Titre: Title:	Modélisation des effets d'historique dans un réacteur à eau pressurisée
Auteur: Author:	Benjamin Toueg
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Toueg, B. (2011). Modélisation des effets d'historique dans un réacteur à eau pressurisée [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/561/

UNIVERSITÉ DE MONTRÉAL

MODÉLISATION DES EFFETS D'HISTORIQUE DANS UN RÉACTEUR À EAU PRESSURISÉE.

BENJAMIN TOUEG
DÉPARTEMENT DE GÉNIE PHYSIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉNERGÉTIQUE)
AVRIL 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MODÉLISATION DES EFFETS D'HISTORIQUE DANS UN RÉACTEUR À EAU PRESSURISÉE.

présenté par : TOUEG Benjamin
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. KENNEDY, Gregory, Ph.D., président.
M. KOCLAS, Jean, Ph.D., membre et directeur de recherche.
M. HÉBERT, Alain, D.Ing., membre et codirecteur de recherche.
M. GIRARDI, Enrico, Ph.D., membre.

À mes parents,
à ma sœur,
à Milenda...

REMERCIEMENTS

Jean Koclas qui m'a accepté dans le programme de génie nucléaire. Alain Hébert pour avoir rendu possible cette collaboration avec EDF. Tanguy Coureau, pour son expertise et son soutien. Yves Dutheillet et le groupe I27/SINETICS pour leur accueil : David Couyras pour son expertise sur COCAGNE, Matthieu Guillo et Yann Pora pour leur aide technique, Fabrice Hoareau pour son aide sur COCAGNE, Alexandre Bonne pour sa bonne humeur, Joël Le Mer, Nicolas Jourdheuil pour les parties de squash. À Poly, je remercie Altan Muftuoglu pour le squash et Nicolas Martin pour son aide sur DRAGON et son soutien. Je remercie aussi Élizabeth Varin pour ses conseils sur la présentation des résultats de ce mémoire. Milenda Zihouf avec qui j'ai partagé cette expérience à Montréal.

RÉSUMÉ

Les codes de physique des réacteurs DRAGON4 et DONJON4, respectivement dédiés à l'étude neutronique des réseaux et des cœurs entiers, ont été conçus par l'Institut de Génie Nucléaire (IGN) pour simuler des CANDU et des Advanced CANDU Reactor. Le code DRAGON4 peut, en théorie, s'appliquer aux Réacteurs à Eau légère Pressurisée (REP) mais le code DONJON4 requiert un réacteur de type CANDU et doit être adapté pour traité les REP.

La physique des REP est différente de celle des réacteurs CANDU. La présence d'eau légère, la géométrie, le type de combustible ainsi que le type de rechargement affectent le comportement du réacteur lorsque certains de ses paramètres sont modifiés.

Électricité de France (EDF), dont le parc nucléaire est consititué de REPs, a sollicité la création d'une simulation avec "effets d'historique", à l'aide des codes de l'IGN, sur une campagne de production représentative d'un réacteur de sa flotte.

Le premier objectif de la maîtrise a été la mise à niveau d'un calcul DRAGON3/DONJON3 vers DRAGON4/DONJON4 afin de le comparer avec un calcul DRAGON4/COCAGNE d'EDF. Cette comparaison inédite (des modifications ont été apportées aux codes de l'IGN afin de les adapter aux particularités des REPs) a d'une part, permis la validation de DONJON4, et d'autre part, aboutit à une réflexion sur la modélisation des mécanismes de contre-réaction dans les codes de cœur concernés.

Le deuxième objectif a été la mise en place d'un schéma de calcul avec "effets d'historique". EDF a fourni du temps sur un cluster de calcul pour la simulation, et DONJON4 a été parallélisé pour tirer parti de ces ressources. Une simulation de l'exploitation d'un REP a été effectuée en utilisant un scénario pré-établi, correspondant à une campagne de production représentative d'un réacteur de la flotte EDF. Celle-ci a été comparée au schéma de calcul de la première partie et les écarts ont été analysés.

ABSTRACT

The neutronic codes DRAGON4 and DONJON4, respectively dedicated to lattices and cores calculations, have been designed by the Institute of Nuclear Engineering (IGN) to simulate CANDU and Advanced CANDU reactors. DRAGON4 code can theoretically be applied to pressurized water reactors (PWRs) but DONJON4 code is more oriented towards CANDU reactors and requires some adaptation to be applied to PWR.

PWR physics is different from that of a CANDU reactor. The presence of light water, the geometry, the fuel type and the refuelling type affect the behavior of the reactor when some of these parameters are changed.

Électricité de France (EDF), which has a reactor fleet of PWRs, solicited the creation of a "history-based" simulation over a campaign length of a typical reactor of theirs, using IGN's codes.

The first objective of this work was to update a PWR calculation from DRAGON3 / DONJON3 to DRAGON4 / DONJON4 and to compare it with a DRAGON4 / COCAGNE calculation - COCAGNE being a core code made by EDF. This new and unique comparison (changes were made to IGN's codes to fit the specificities of PWRs) has enabled the validation of DONJON4, and has also led to a reflection on counter-reaction mechanisms in core calculations.

The second objective was to establish a "history based" calculation scheme. EDF has provided some calcultation on its cluster Clamart2, and DONJON4 has been parallelized to take advantage of these resources. A campaign length simulation of a PWR was conducted using a pre-established scenario. It was compared with the calculation scheme of the first part and differences have been analyzed.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

CANDU	CANada Deuterium Uranium
CP	Collision Probability
EDF	Électricité de France
GCC	GNU Compiler Collection
GPL	GNU General Public License
IGN	Institut de Génie Nucléaire
MOC	Method of Characteristics
MPI	Message Passing Interface
PBS	Portable Batch System
PWR	Pressurized Water Reactor
REP	Réacteur à Eau Pressurisée
SPH	Superhomogénéisation

CHAPITRE 1

INTRODUCTION

Libérer l'énergie nucléaire que contient l'uranium se fait par une réaction en chaîne de fission dans laquelle les neutrons jouent le rôle de projectile pour scinder les noyaux lourds. C'est l'équation de transport de Boltzmann qui est utilisée pour modéliser l'évolution de la population de neutrons. L'outil principal qu'utilise l'ingénieur R&D pour obtenir la distribution de puissance est la simulation neutronique. Il a besoin que celle-ci soit rapide, précise et robuste. Les capacités de calcul actuelles ne nous permettent pas de résoudre l'équation de transport sur un réacteur complet avec tous les détails des plans de conception. Des approximations et des simplications bien choisies sont introduites dans la simulation jusqu'à l'obtention de ce que l'on appelle un schéma de calcul.

1.1 Définitions et concepts de base

Les approximations diffèrent selon le type de réacteur à étudier, donc les schémas de calculs diffèrent eux aussi. Les réacteurs qui nous intéressent dans ce mémoire sont les Réacteurs à Eau légère Pressurisé (REP). Ils représentent la majorité des réacteurs installés dans le monde, et c'est l'unique type de réacteur exploité par Électricité de France (EDF) en 2009/2010.

1.1.1 Réacteur à Eau légère Pressurisée

Nous étudierons un cœur de REP typique d'EDF constitué 157 assemblages de section carrée (environ 20 cm sur 20 cm) et de 4m de hauteur (voir figure 1.1). Chaque assemblage est constitué de 289 tubes de zircaloy, faisant toute la hauteur de l'assemblage, dont 264 sont des crayons combustibles (voir figure 1.2). L'eau sous pression (155 bars) entre en partie basse du cœur à 280°C environ et en sort en partie haute à plus de 320°C.

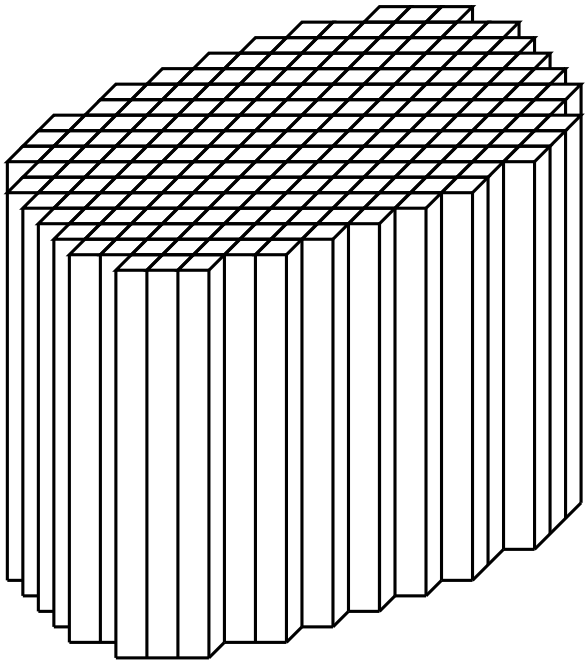


Figure 1.1 REP : disposition des 157 assemblages dans le cœur

La durée de d'exploitation d'un réacteur REP est d'environ 50 ans, ce qui est d'un ordre de grandeur supérieur à son autonomie. Ceci signifie qu'il va falloir recharger le réacteur en combustible neuf. Au cours de la procédure de rechargement certains assemblages trop usés sortent du réacteur, d'autres sont déplacés et de nouveaux sont intégrés dans le cœur.

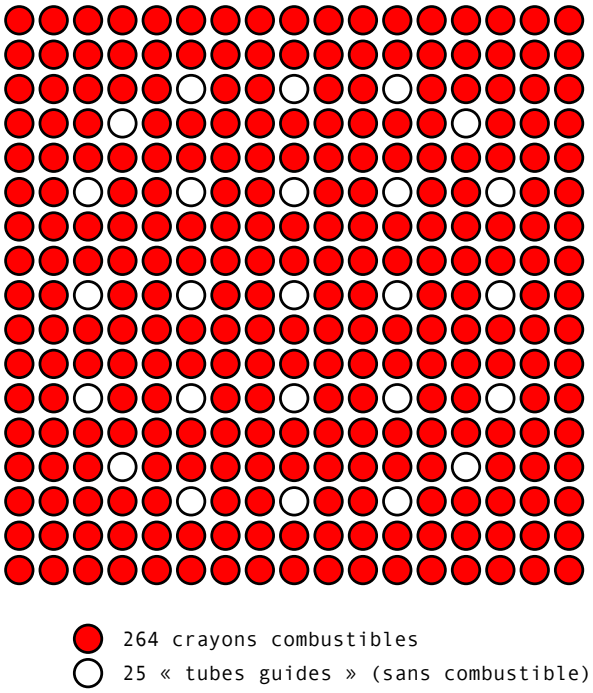


Figure 1.2 La constitution d'un assemblage de combustible

1.1.2 Longueur de campagne

Une longueur de campagne est la durée pendant laquelle un réacteur est exploité entre deux rechargements. Au début de la période d'exploitation, le combustible dans le réacteur est surcritique. L'excès de réactivité est compensé par l'ajout d'un poison à neutron (le bore) dans le liquide refroidissement (l'eau). Au fur et à mesure que le combustible brûle, sa réactivité baisse, et l'on diminue la concentration de bore. Une fois qu'il n'est plus possible de retirer du bore, le réacteur est arrêté pour rechargement.

1.2 Éléments de la problématique

Un schéma de calcul est fait d'un calcul d'assemblage et d'un calcul de cœur. Dans le premier le comportement des neutrons est régi par l'équation de transport à l'échelle de l'assemblage et est modélisé par un code de réseau tel DRAGON4. Lors du second calcul,

le comportement des neutrons est régi par l'équation de la diffusion à l'échelle du cœur (DONJON4, COCAGNE).

1.2.1 Calcul d'assemblage

Le calcul d'assemblage constitue la première étape, et sert à produire une bibliothèque de sections efficaces correspondant à un milieu homogène équivalent à l'assemblage. Cette bibliothèque est paramétrée suivant les propriétés de certains matériaux de l'assemblage.

Dans DRAGON4, un assemblage est modélisé avec ses 289 crayons (figure 1.2) sous des conditions aux limites de réflexion. La bibliothèque résultante est ensuite utilisée dans DONJON4 (ou COCAGNE), où les assemblages sont modélisés par un milieu homogène.

1.2.2 Calcul de cœur classique

La géométrie du calcul de cœur est constituée de volumes élémentaires de combustible (cellules) possèdant leurs propres température, densité, concentration et burnup (ou taux de combustion).

Pour un calcul de cœur classique, la bibliothèque générée par le code d'assemblage est une donnée d'entrée qu'il n'est pas possible de modifier. Elle contient les sections efficaces générées pour un assemblage « type » et pour un nombre fini de valeurs. Le code de cœur utilise cette bibliothèque pour décrire tous ses assemblages et interpole les sections de cette bibliothèque pour réaliser son calcul.

1.2.3 Calcul d'historique

Une alternative au calcul de cœur classique est le calcul de cœur avec effet d'historique (Varin et Marleau (2006)). Un calcul d'historique consiste à réaliser un couplage entre le code de cœur et le code d'assemblage. Contrairement au calcul classique où DONJON4 interpole les sections dans une bibliothèque, le calcul d'historique commande à la volée au code d'assemblage les sections aux valeurs désirées pour chaque cellule de la géométrie. Chaque assemblage du cœur est brûlé dans DRAGON en utilisant les conditions exactes de son historique, mais avec une approximation de mode fondamental.

1.2.4 Influence du chemin

La création de la bibliothèque se fait en brûlant un assemblage à température et densité nominales. Pour ajouter à la bibliothèque un point de burnup B et de température T, DRAGON4 part du point $(B, T_{nominale})$ pour arriver au point (B, T). A priori, on pourrait arriver au point (B, T) en partant d'un point (B', T), ce qui ne changerait pas les coordonnées

de la bibliothèque. Mais les deux chemins diffèrent et cela ne correspond pas physiquement à la même situation (exemple sur la figure 1.3), ce qui a un impact sur les sections efficaces utilisées par le code de cœur.

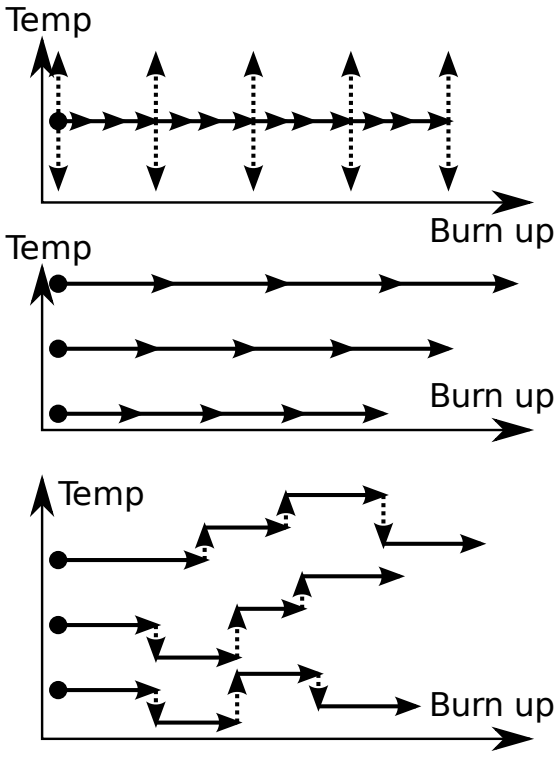


Figure 1.3 Example de coordonnées (pointes des flèches) atteintes par des chemins différents

Ce travail de recherche vise à répondre aux question suivantes : le calcul d'assemblage génère-t-il une bibliothèque suffisamment universelle pour convenir à toutes les cellules dans le calcul classique ? Un calcul historique apporte-t-il un gain de précision dans le calcul ? Est-il possible de le mettre en place dans DONJON4 ? Pour quel temps de calcul ?

1.3 Objectifs de recherche

Les objectifs sont les suivants :

- Implémenter le module TINST: dans DONJON4, afin de pouvoir faire brûler le combustible. Le nom TINST: a été choisi car le calcul de combustion est instantané par opposition au module TAVG: qui utilise un modèle de moyenne temporelle adapté aux CANDU.
- Valider DONJON4 en le comparant à COCAGNE sur un calcul de cœur classique.
- Implémenter les modules DRVMPI: et SNDMPI: pour ajouter à la GANLIB la possibilité d'utiliser MPI.
- Ajuster le code source pour permettre une compilation en 64-bits.
- Effectuer un calcul d'historique en utilisant MPI en 64-bits avec DONJON4 sur un cluster d'EDF.
- Analyser les résultats du calcul.

1.4 Plan du mémoire

Dans une première partie nous présenterons les éléments théoriques nécessaires à la compréhension des calculs d'assemblage et de cœur ainsi que les codes informatiques utilisés.

Ensuite nous étudierons le schéma de calcul mis en place dans le cas classique, de l'assemblage jusqu'à une longueur de campagne complète en passant par toutes les étapes intermédiaires permettant la validation de DONJON4.

La dernière partie sera consacrée à la mise en place du calcul d'historique en parallèle et à son analyse.

CHAPITRE 2

ÉLÉMENTS DE NEUTRONIQUE

2.1 Présentation de l'équation de transport

L'approche issue de la mécanique statistique suppose que chaque neutron évolue dans un espace à 7 dimensions comprenant :

- 3 coordonnées spatiales ;
- 3 coordonnées angulaires ;
- 1 dimension temporelle.

On utilise couramment la notation suivante (Hébert (2009)) :

- $\vec{r}$ pour la variable de position dans l'espace.
- $\overrightarrow{V_n}$ variable de vitesse décomposée en $V_n = \left|\overrightarrow{V_n}\right|$ et $\widehat{\Omega} = \dfrac{\overrightarrow{V_n}}{V_n}$.
- t variable temporelle.

La densité neutronique est représentée par la distribution $n(\vec{r}, V_n, \widehat{\Omega}, t)$,
tel que $n(\vec{r}, V_n, \widehat{\Omega}, t) d^3r dV_n d^2\Omega$ représente le nombre de neutrons à l'instant t dans l'hypervolume $d^3r dV_n d^2\Omega$, qui correspond à l'élément de volume d^3r autour du point $\vec{r}$, l'élément de vitesse dV_n autour de V_n, et dans l'élément d'angle solide $d^2\Omega$ autour de $\widehat{\Omega}$. On peut remarquer que $n(\vec{r}, V_n, \widehat{\Omega}, t)$ est une distribution par rapport aux variables $\vec{r}$, V_n et $\widehat{\Omega}$ mais une fonction par rapport au temps t.
On définit à partir de la densité neutronique le flux neutronique tel que :

$$\phi(\vec{r}, V_n, \widehat{\Omega}, t) = V_n n(\vec{r}, V_n, \widehat{\Omega}, t) \tag{2.1}$$

Le flux neutronique tel que défini précédemment est utilisé comme inconnue dans l'équation de transport car sa connaissance facilite par la suite le calcul des taux de réactions. Ainsi, sa définition est d'ordre mathématique et non physique.
On utilisera tout au long de ce travail la forme dite intégro-différentielle simplifiée de l'équation de transport de Boltzmann :

$$\widehat{\Omega} \cdot \nabla\phi(\vec{r}, V_n, \widehat{\Omega}) + \Sigma(\vec{r}, V_n)\phi(\vec{r}, V_n, \widehat{\Omega}) = Q(\vec{r}, V_n, \widehat{\Omega}) \tag{2.2}$$

Avec :

- $\vec{r}$ variable d'espace,

- $\widehat{\Omega} = (\sqrt{(1-\mu^2)}cos\phi, \sqrt{(1-\mu^2)}sin\phi, \mu)$ variable angulaire,
- V_n variable d'énergie.
- $\phi(\vec{r}, V_n, \widehat{\Omega})$ représente le flux neutronique dans l'hypervolume associé,
- $\widehat{\Omega} \cdot \nabla\phi(\vec{r}, V_n, \widehat{\Omega})$ représente le terme de neutrons sortant,
- $\Sigma(\vec{r}, V_n)\phi(\vec{r}, V_n, \widehat{\Omega})$ représente le terme de fuite par collision des neutrons,
- $Q(\vec{r}, V_n, \widehat{\Omega})$ représente l'ensemble des termes de source des neutrons.

2.2 Obtention de l'équation de transport

L'équation de transport traduit un bilan de neutrons dans l'élément d'hypervolume défini précédemment $d^3rd^2\Omega dV_n$ autour de $\{\vec{r}, V_n, \widehat{\Omega}\}$ dans un intervalle de temps Δt :

$$\begin{aligned} \text{Variation du nombre de neutrons} = & - \text{ bilan de neutrons sortis de } d^3r \\ & - \text{ neutrons sortis par collision} \\ & + \text{ neutrons créés} \end{aligned} \tag{2.3}$$

On détaille chaque terme :

- La variation du nombre de neutrons dans l'intervalle de temps Δt dans l'hypervolume $d^3rd^2\Omega dV_n$ est donnée par :

$$n(\vec{r}, V_n, \widehat{\Omega}, t + \Delta t) - n(\vec{r}, V_n, \widehat{\Omega}, t) \tag{2.4}$$

- Les neutrons perdus par collision pendant l'intervalle de temps Δt s'écrivent :

$$\Sigma(\vec{r}, V_n) \left[\phi(\vec{r}, V_n, \widehat{\Omega}, t)\right] \Delta t \tag{2.5}$$

  avec $\Sigma(\vec{r}, V_n)$ section efficace totale du milieu.
- Le bilan de neutrons sortant de d^3r se met sous la forme :

$$\overrightarrow{\nabla} \cdot \widehat{\Omega}\phi(\vec{r}, V_n, \widehat{\Omega}, t)\Delta t = \widehat{\Omega} \cdot \overrightarrow{\nabla}\phi(\vec{r}, V_n, \widehat{\Omega}, t)\Delta t \tag{2.6}$$

- on définit le terme source $Q(\vec{r}, V_n, \widehat{\Omega}, t)$ tel que

$$Q(\vec{r}, V_n, \widehat{\Omega}, t)\Delta t \tag{2.7}$$

  représente le nombre de neutrons créés pendant l'intervalle de temps Δt dans l'élément $d^3rd^2\Omega dV_n$.

On peut désormais établir le bilan de la population neutronique dans l'élément d'hypervolume $d^3r d^2\Omega dV_n$ autour de $\{\vec{r}, V_n, \widehat{\Omega}\}$ dans un intervalle de temps Δt :

$$\begin{aligned}\frac{n(\vec{r}, V_n, \widehat{\Omega}, t+\Delta t) - n(\vec{r}, V_n, \widehat{\Omega}, t)}{\Delta t} = &- \Sigma(\vec{r}, V_n)\phi(\vec{r}, V_n, \widehat{\Omega}, t) \\ &- \widehat{\Omega} \cdot \nabla\phi(\vec{r}, V_n, \widehat{\Omega}, t) \\ &+ Q(\vec{r}, V_n, \widehat{\Omega}, t)\end{aligned} \tag{2.8}$$

En faisant tendre Δt vers 0, on obtient la forme *différentielle* de l'équation de transport :

$$\frac{1}{V_n}\frac{\partial}{\partial t}\phi(\vec{r}, V_n, \widehat{\Omega}, t) + \overrightarrow{\nabla} \cdot \widehat{\Omega}\phi(\vec{r}, V_n, \widehat{\Omega}, t) + \Sigma(\vec{r}, V_n)\phi(\vec{r}, V_n, \widehat{\Omega}, t) = Q(\vec{r}, V_n, \widehat{\Omega}, t) \tag{2.9}$$

Avec l'égalité

$$\overrightarrow{\nabla} \cdot \widehat{\Omega}\phi(\vec{r}, V_n, \widehat{\Omega}, t) = \widehat{\Omega} \cdot \overrightarrow{\nabla}\phi(\vec{r}, V_n, \widehat{\Omega}, t), \tag{2.10}$$

On obtient la forme suivante pour l'équation linéaire de Boltzmann :

$$\frac{1}{V_n}\frac{\partial}{\partial t}\phi(\vec{r}, V_n, \widehat{\Omega}, t) + \widehat{\Omega} \cdot \overrightarrow{\nabla}\phi(\vec{r}, V_n, \widehat{\Omega}, t) + \Sigma(\vec{r}, V_n)\phi(\vec{r}, V_n, \widehat{\Omega}, t) = Q(\vec{r}, V_n, \widehat{\Omega}, t) \tag{2.11}$$

En régime stationnaire, l'équation devient :

$$\widehat{\Omega} \cdot \overrightarrow{\nabla}\phi(\vec{r}, V_n, \widehat{\Omega}) + \Sigma(\vec{r}, V_n)\phi(\vec{r}, V_n, \widehat{\Omega}) = Q(\vec{r}, V_n, \widehat{\Omega}) \tag{2.12}$$

2.3 Traitement du terme de source

Dans l'équation linéaire stationnaire de Boltzmann, le terme de source $Q(\vec{r}, V_n, \widehat{\Omega})$ englobe toutes les émissions possibles de neutrons (fission, diffusion, source externe). On utilise désormais pour l'énergie la variable $E = \frac{1}{2}mV_n^2$. Sous sa forme intégrale, le terme de source s'écrit :

$$\begin{aligned}Q(\vec{r}, \widehat{\Omega}, E) = &\int_0^\infty dE' \int_{4\pi} d^2\Omega' \Sigma_s(\vec{r}, \widehat{\Omega} \leftarrow \widehat{\Omega}', E \leftarrow E')\phi(\vec{r}, \widehat{\Omega}, E') \\ &+ \frac{1}{4\pi k_{\text{eff}}} Q^{\text{fiss}}(\vec{r}, E)\end{aligned} \tag{2.13}$$

Avec $Q^{\text{fiss}}(\vec{r}, E)$ la source isotropique de fission : on considère qu'elle est indépendante de l'énergie du neutron incident. Elle s'écrit :

$$Q^{\text{fiss}}(\vec{r}, E) = \sum_{j=1}^{J^{\text{fiss}}} \chi_j(E) \int_0^{\infty} dE' \nu_j \Sigma_{f,j}(\vec{r}, E') \phi(\vec{r}, E') \tag{2.14}$$

Les valeurs macroscopiques sont les suivantes :

- Σ_s la section efficace de diffusion (scattering), correspondant au transfert entre les différents groupes d'énergie ;
- k_{eff} est le facteur de multiplication effectif. Si la somme des absorptions et des fuites n'est pas égale à la production de neutrons par la fission, on perd l'état d'équilibre stationnaire. k_{eff} est le facteur par lequel on doit diviser la source de fission pour maintenir cet équilibre.
- J^{fiss} le nombre d'isotopes fissiles ;
- $\Sigma_{f,j}$ section efficace de fission ;
- ν_j le nombre de neutrons secondaires issus de la fission ;
- χ_j le spectre de fission, défini tel que $\chi_j(E)dE$ corresponde à la probabilité pour un neutron émis suite à la fission d'un isotpe j d'avoir une énergie E à dE près.

Avant de résoudre l'équation de transport dans une région V donnée, il convient de s'intéresser aux conditions imposées aux limites de ce domaine ∂V.

2.4 Traitement des conditions aux frontières

Le traitement des conditions aux limites est propre à chaque méthode de résolution. Pour l'instant, de manière générale, on pose $\overrightarrow{N(r_s)}$ le vecteur normal au point $\overrightarrow{r_s}$ appartenant à ∂V.
La résolution de l'équation de Boltzmann sur le domaine imposé V requiert la connaissance du flux $\phi(\vec{r}, V_n, \widehat{\Omega})$ pour $\overrightarrow{\Omega} \cdot N(\overrightarrow{r_s}) < 0$.

- Condition limite d'albédo :
  La condition aux limites d'albédo relie la valeur du flux sortant à celle du flux rentrant. Cette relation s'exprime par :

$$\phi(\vec{r}, V_n, \widehat{\Omega}) = \beta \phi(\vec{r}, V_n, \widehat{\Omega}') \text{ avec } \overrightarrow{\Omega} \cdot N(\overrightarrow{r_s}) < 0 \tag{2.15}$$

  On a noté $\widehat{\Omega}'$ la direction de la particule sortante. L'albédo β peut prendre n'importe

quelle valeur entre 0 et 1, si $\beta = 0$ alors nous sommes dans le cas de conditions de vide aux frontières. Si β prend la valeur 1, il s'agit de conditions de réflexion pure.
Le cas général correspond au cas dit de réflexion spéculaire :

$$\widehat{\Omega} \cdot N(\overrightarrow{r_s}) = -\widehat{\Omega'} \cdot N(\overrightarrow{r_s}) \text{ avec } (\widehat{\Omega} \times \widehat{\Omega'}) \cdot N(\overrightarrow{r_s}) = 0 \tag{2.16}$$

– Conditions de réflexion blanche :
Elle correspond à une condition particulière de réflexion pour laquelle toutes les particules sortant de l'espace V retournent dans V avec une distribution angulaire isotrope. On a alors :

$$\phi(\vec{r}, V_n, \widehat{\Omega}) = 4\pi \int_{\widehat{\Omega'} \cdot N(\overrightarrow{r_s}) > 0} d^2\Omega' \left[\widehat{\Omega'} \cdot N(\overrightarrow{r_s})\right] \phi(\vec{r}, V_n, \widehat{\Omega'}) \text{ avec } \widehat{\Omega'} \cdot N(\overrightarrow{r_s}) < 0 \tag{2.17}$$

– Conditions de translation ou conditions aux frontières périodique :
Ce type de condition correspond au cas d'un réseau périodique où le flux sur une frontière est égal au flux sur une autre frontière parallèle au domaine.

$$\phi(\overrightarrow{r_s}, V_n, \widehat{\Omega}) = \phi(\overrightarrow{r_s} + \Delta r, V_n, \widehat{\Omega}) \tag{2.18}$$

avec Δr le pas du réseau.

2.5 L'approche multigroupe

La résolution de l'équation de transport par le biais de méthode déterministe implique un traitement spécial de la dépendance en énergie. L'approche multigroupe consiste à subdiviser le domaine d'énergie en G groupes dans lesquels les neutrons seront considérés comme monocinétiques. Les quantités dépendant de l'énergie seront ensuite condensées sur chaque groupe.
On utilise de manière courante la variable léthargie $u = ln(E_0/E)$ sur le domaine énergétique $[0, E_0]$, de tel sorte que :

$$W_g = \{u; u_{g-1} \leq u \leq u_g\} = \{E; E_g \leq E \leq E_{g-1}\}; \qquad g = 1, G \tag{2.19}$$

avec $u_g = ln(E_0/E_g)$ et $u_0 = 0$. Ainsi le spectre d'énergie est divisé en G groupes $]E_g, E_{g-1}[$ avec $g = [1, G]$. Les groupes de plus hautes énergies se retrouvent donc en premier, $E_1 > E_2 ... > E_G$.

On peut dès lors exprimer l'équation de transport en formalisme multi-groupe :

$$\widehat{\Omega} \cdot \overrightarrow{\nabla} \phi^g(\vec{r}, \widehat{\Omega}) + \Sigma^g(\vec{r}) \phi^g(\vec{r}, \widehat{\Omega}) = Q^g(\vec{r}, \widehat{\Omega}) \tag{2.20}$$

Avec désormais la relation suivante pour le terme source :

$$\begin{aligned} Q^g(\vec{r}, \widehat{\Omega}) = & \sum_{g'=1}^{G} \int_{4\pi} d^2\Omega' \Sigma_s^{g \leftarrow g'}(\vec{r}, \widehat{\Omega} \leftarrow \widehat{\Omega}') \phi^{g'}(\vec{r}, \widehat{\Omega}') \\ & + \frac{1}{4\pi k_{\text{eff}}} \sum_{j=1}^{J^{\text{fiss}}} \chi_j^g(\vec{r}) \sum_{g'=1}^{G} \nu_j \Sigma_{f,j}^{g'}(\vec{r}) \phi^{g'}(\vec{r}) \end{aligned} \tag{2.21}$$

Si on appelle s l'abscisse curviligne, cette équation peut se mettre sous la forme suivante qu'on appelle la forme caractéristique de l'équation de transport :

$$\frac{d}{ds} \phi^g(\vec{r} + s\widehat{\Omega}, \widehat{\Omega}) + \Sigma_g(\vec{r} + s\widehat{\Omega}) \phi^g(\vec{r} + s\widehat{\Omega}, \widehat{\Omega}) = Q^g(\vec{r} + s\widehat{\Omega}, \widehat{\Omega}) \tag{2.22}$$

En posant le chemin optique $\tau^g(s) = \int_0^s ds' \Sigma^g(\vec{r} + s'\widehat{\Omega})$, la forme intégrale en domaine infini de l'équation (2.22) s'écrit :

$$\phi^g(\vec{r}, \widehat{\Omega}) = \int_0^{\infty} ds \; e^{-\tau^g(s)} Q^g(\vec{r} - s\widehat{\Omega}) \tag{2.23}$$

2.6 Méthodes de résolution de l'équation du transport

Nous allons présenter les principales méthodes de résolution de l'équation du transport disponible dans DRAGON.

2.6.1 Méthodes des probabilités de collision

Présentation des matrices p_{ij}

La méthode des probabilités de collision (CP) est issue de la discrétisation spatiale de la forme intégrale de l'équation de transport multigroupe (2.23). On intègre cette équation par

rapport à son angle solide et on introduit le changement de variable $\overrightarrow{r'} = \vec{r} - s\widehat{\Omega}$:

$$\begin{aligned}\phi^g(\vec{r}) = \int_{4\pi} d^2\Omega \phi^g(\vec{r},\widehat{\Omega}) &= \int_{4\pi} d^2\Omega \int_0^\infty ds\ e^{-\tau^g(s)} Q^g(\vec{r} - s\widehat{\Omega}) \\ &= \int_\infty d^3r' \frac{e^{-\tau^g(s)}}{s^2} Q^g(\overrightarrow{r'})\end{aligned} \tag{2.24}$$

En général, on utilise cette méthode pour un espace pavé à l'infini de cellules unitaires. On découpe chacune de ces cellules en régions V_i. On utilise la notation V_i^∞ pour représenter toutes les régions V_i sur le pavage infini. On développe la section efficace de diffusion en polynôme de Legendre et on ne conserve que le premier terme, on a : $\Sigma^{g\leftarrow g'}_{s,i,\widehat{\Omega}\leftarrow\widehat{\Omega}'} = \frac{\Sigma^{g\leftarrow g'}_{s,0,i}}{4\pi}$. On suppose que la source de neutrons Q_i^g est uniforme sur la région V_i (ce qui explique la nécessité de réaliser une discrétisation fine de la cellule).

En multipliant l'équation (2.24) par $\Sigma^g(\vec{r})$ et en intégrant sur chaque région V_i, on trouve :

$$\int_{V_j} d^3r \Sigma^g(\vec{r})\phi^g(\vec{r}) = \int_{V_j} d^3\Sigma^g(\vec{r}) \sum_i Q_i^g \int_{V_i^\infty} d^3r' \frac{e^{-\tau^g(s)}}{s^2} \tag{2.25}$$

Avec

$$Q_i^g = \sum_{g'=1}^{G} \frac{\Sigma^{g\leftarrow g'}_{s,0,i}}{4\pi}\phi_i^{g'} + \frac{1}{4\pi k_{\text{eff}}} \sum_{j=1}^{J^{\text{fiss}}} \chi_j^g \sum_{g'=1}^{G} \Sigma^{g'}_{f,j}\phi_i^{g'} \tag{2.26}$$

L'équation (2.25) peut être simplifiée sous la forme suivante :

$$\phi_j^g = \frac{1}{V_j\Sigma_j^g} \sum_i Q_{i,j} V_i P_{ij}^g \tag{2.27}$$

où

$$\phi_j^g = \frac{1}{V_j} \int_{V_j} d^3r \phi^g(\vec{r}) \tag{2.28}$$

$$\Sigma_j^g = \frac{1}{V_j\phi_j^g} \int_{V_j} d^3r \Sigma^g(\vec{r})\phi^g(\vec{r}) \tag{2.29}$$

$$P_{ij}^g = \frac{1}{4\pi V_i} \int_\infty d^3r' \int_{V_j} d^3r \Sigma^g(\vec{r}\frac{e^{-\tau^g(s)}}{s^2}) \tag{2.30}$$

P_{ij}^g est la probabilité pour un neutron créé dans une des régions V_i du réseau de réaliser sa première collision dans la région V_j du motif étudié, c'est ce que nous appelons une probabilité de collision (CP). On considère en général que les sections efficaces sont constantes dans

chaque région V_j , on peut alors utiliser les CP réduites p_{ij}^g telles que :

$$p_{ij}^g = \frac{P_{ij}^g}{\Sigma_j^g} = \frac{1}{4\pi V_i} \int_\infty d^3r' \int_{V_j} d^3r \frac{e^{-\tau_g(s)}}{s^2} \tag{2.31}$$

Les CP réduites ont les propriétés suivantes :

- réciprocité : $p_{ij}^g V_i = p_{ji}^g V_j$
- conservation : $\sum_j p_{ij}^g \Sigma_j^g = 1$

On peut donc écrire l'équation (2.27) de la manière suivante :

$$\phi_i^g = \sum_j Q_j^g p_{ij}^g \tag{2.32}$$

La première étape d'un calcul neutronique utilisant cette méthode consiste donc à calculer les CP, cette étape est présentée dans la section 2.6.1. Une fois que les CP sont connues, le flux intégré est calculé par itération à partir des équations (2.27) et (2.32).

On peut ramener ces équations sous la forme suivante :

$$\mathbf{\Phi}_g = \mathcal{W}_g \mathbf{Q}_g^* \tag{2.33}$$

avec

$$\begin{aligned}
&\mathbf{\Phi}_g = \{\phi_i^g; \forall i\} \\
&\mathbf{Q}_g^* = \left\{ \sum_{g' \neq g} \Sigma_{s,0,i}^{g \leftarrow g'} \phi_i^{g'} + \frac{1}{k_{\text{eff}}} Q_i^{\text{fiss } g}; \forall i \right\} \\
&\mathcal{W}_g = [\mathcal{I} - \mathcal{P}_g \mathcal{S}_{s,0}^{g \leftarrow g}]^{-1} \mathcal{P}_g \\
&\mathcal{P}_g = \{p_{ij}^g; \forall i, j\} \\
&\mathcal{S}_{s,0}^{g \leftarrow g} = diag\{\Sigma_{s,0,i}^{g \leftarrow g'}\}
\end{aligned} \tag{2.34}$$

Deux procédés itératifs sont en général superposés. Une itération interne est effectuée sur les sources de diffusion jusqu'à l'obtention d'un flux multigroupe. Une itération externe appelée itération de puissance est effectuée pour converger sur le k_{eff} ou sur les fuites (voir section 2.8).

En règle générale, la limitation technique de la méthode des CP est le nombre n de régions considérées. En effet, l'inversion dematrices de dimension $n \times n$ est un processus non

linéaire et l'augmentation du nombre de régions fait exploser le temps de calcul et l'espace mémoire nécessaire au stockage des matrices. On considère en général que 1000 à 5000 régions (dépendamment du nombre de groupes d'énergie) est le maximum que l'on peut utiliser dans le cadre d'un calcul multigroupe.

Calcul des matrices p_{ij}, définitions des lignes d'intégration

La matrice des CP est généralement évaluée en deux étapes :

- On définit des lignes d'intégration de manière à représenter suffisamment de trajectoires de neutrons : On décompose le domaine angulaire en une série de directions et associons à chaque direction $\widehat{\Omega}_m$ un poids ω_m de telle manière que $\int_0^{4\pi} d^2\Omega = \sum_m \omega_m \widehat{\Omega}_m = 4\pi$. On choisit ensuite un plan $\Pi_{\widehat{\Omega}_m}$ normal pour chaque direction et divise ce plan en une grille uniforme en générant ainsi une série de points d'intégration $\overrightarrow{p}_{m,n}$ avec un poids $\Pi_{m,n}$. L'ensemble de l'espace est ainsi discrétisé en lignes et en points d'intégration, ce qu'on appelle le "tracking". Il faut alors identifier les intersections entre le tracking et les régions du problème à traiter de telle manière que nous puissions calculer explicitement les chemins optiques.
- Une intégration numérique des probabilités de collision est réalisée en utilisant le tracking et les sections efficaces totales dans chaque région. L'équation (2.31) peut alors s'écrire :

$$p_{ij} = \frac{1}{\Sigma_i \Sigma_j V_i} \sum_m \omega_m \sum_n \Pi_{m,n} \tag{2.35}$$

$$\sum_k \delta_{i,V_k} \sum_h \delta_{j,V_h} [1 - e^{-\Sigma_i L_k}] e^{-\tau_{k,h}} [1 - e^{-\Sigma_j L_h}] \tag{2.36}$$

avec L_k la distance traversée à l'intérieur de la région V_k par un neutron généré au point $\overrightarrow{p}_{m,n}$ et se déplaçant dans la direction $\widehat{\Omega}_m$.

2.6.2 Méthodes des courants d'interface

Il existe une méthode approchée plus rapide en temps CPU appelée méthode des courants d'interface (Sanchez et McCormik (1982)). Elle consiste à calculer séparément les matrices des CP de chaque cellule. On peut ensuite reconstruire le flux détaillé à partir des courants d'interface sur chaque cellule. Le flux angulaire sortant est exprimé selon une double expansion P_n. De la même manière que précédemment, on définit des probabilités de collision :

- p_{ij} probabilité pour un neutron créé uniformément et isotropiquement dans la région i d'avoir sa première collision dans la région j sans sortir de la cellule.

- $p^{(\rho)}_{S_\alpha j}$ probabilité pour un neutron entrant uniformément dans la cellule avec la distribution angulaire $\psi_\rho(\widehat{\Omega}, \overrightarrow{N}^-)$ d'avoir sa première collision dans la région j sans sortir de la cellule.
- $p^{(\nu)}_{iS_\beta j}$ probabilité pour un neutron créé uniformément et isotropiquement dans la région i de sortir de la cellule avec la distribution angulaire $\psi_\nu(\widehat{\Omega}, \overrightarrow{N}^+)$.
- $p^{(\rho\nu)}_{S_\alpha S_\beta}$ probabilité pour un neutron entrant uniformément dans la cellule avec la distribution angulaire $\psi_\rho(\widehat{\Omega}, \overrightarrow{N}^-)$ de sortir de la cellule avec la distribution angulaire $\psi_\nu(\widehat{\Omega}, \overrightarrow{N}^+)$.

À partir de ces grandeurs, on obtient le même type d'équations que les équations (2.26) et (2.32) mais sur un domaine spatial bien plus réduit ce qui accélère le calcul. Notons que dans le cas d'un système fini (un assemblage 17×17 avec des réflexions sur ses bords par exemple), la méthode CP utilise le même formalisme que présenté ici mais les surfaces ne sont pas celles des cellules unitaires, ce sont celles de l'ensemble de l'espace étudié.

2.6.3 Méthodes des caractéristiques

La méthode des caractéristiques (MOC) (Askew (1972)) est basée sur la discrétisation de la forme caractéristique de l'équation de transport (équation (2.22)). Elle s'appuie sur un calcul itératif du flux neutronique par la résolution de l'équation de transport sur des trajectoires traversant entièrement le domaine. Cette méthode est très proche de la méthode des CP. Elle utilise notamment la même procédure pour générer des lignes d'intégration. On suppose que le segment k passe à travers la région j, la relation locale entre les flux angulaires entrant et sortant est :

$$\Phi^{\text{out}}_j(k) = \Phi^{\text{in}}_j(k) + \left[\frac{Q_j}{\Sigma_j} - \Phi^{\text{in}}_j(k)\right](1 - e^{-\Sigma_j L_k}) \tag{2.37}$$

Le flux scalaire moyen est alors calculé en sommant sur toutes les directions et sur tous les points du tracking. On a donc :

$$\Phi_j = \frac{Q_j}{\Sigma_j} + \sum_m \omega_m \sum_n \Pi_{m,n} \sum_k \delta j V_k (\Phi^{\text{in}}_j(k) - \Phi^{\text{out}}_j(k)) \tag{2.38}$$

Sur les frontières, les conditions de réflexions sont traitées par itération entre les courants entrant et sortant.

La méthode MOC a pour avantage de résoudre le même type d'équation que celle des CP mais avec une matrice de dimension de l'ordre de $n + l$ au lieu de n^2 (n étant le nombre de régions et l le nombre de surfaces), de plus la méthode MOC permet plus simplement de tenir

compte de l'anisotropie du flux. Cependant cet avantage est accompagné de lourds défauts pratiques comme la nécessité de relire entièrement le fichier de tracking à chaque itération interne, ce qui implique la nécessité d'utiliser des méthodes d'accélération efficaces pour que cette méthode soit compétitive (Le Tellier (2006)).

2.6.4 Méthodes stochastiques

Les méthodes stochastiques de résolution de l'équation de transport (Kalos et Whitlock (1986)) reposent sur une approche très différentes des deux méthodes précé-demment présentées. Elle repose sur la méthode de Monte-Carlo : les "vies" de plusieurs millions de neutrons sont simulées en tenant compte de leurs interactions par l'utilisation d'une représentation exacte de la géométrie et d'une représentation continue en énergie des sections efficaces.

Cette méthode est dites stochastique car elle repose pratiquement sur la génération de nombres aléatoires pour rendre compte du comportement statistique des interactions. Cette méthode donne des résultats exacts dans la mesure où la géométrie et les interactions sont correctement simulées et où le nombre d'historiques de neutrons est suffisant. Ce dernier doit être très important ce qui entraîne que les temps de calcul des codes stochastiques sont souvent très longs. Ils sont donc généralement utilisés pour des calculs de référence.

2.7 Modèles d'auto-protection des résonances

Les isotopes lourds, de par la structure nucléaire de leur noyau, présentent pour certaines énergies très localisées du neutron incident de très fortes probabilités d'interaction. Ceci entraîne la nécessité d'une étape de calcul supplémentaire lorsque celui-ci est réalisé dans le formalisme multigroupe usuel à un nombre de groupes faible (entre 50 et 300) pour le traitement de la variable énergétique (Reuss et Coste-Delclaux (2003)). Comme nous l'avons décrit à la section 2.5, la dépendance énergétique des diverses quantités est représentée par des fonctions constantes par morceaux. Or, dans une zone d'énergie correspondant à une résonance, le flux neutronique est largement atténué. Par conséquent, un calcul avec une valeur moyenne de la section efficace dans cette zone (pondérée par un flux qui ne présente pas cette dépression) conduit à une large surestimation du taux de réaction associé. C'est ce qu'on appelle le phénomène d'auto-protection des résonances. Le calcul d'auto-protection est une étape de condensation qui consiste à calculer des estimés des taux de réaction moyens et des flux moyens pour chaque isotope résonnant et pour chaque groupe d'énergie qui présente des résonances de manière à obtenir des sections efficaces dites auto-protégées pour le calcul de flux multigroupe (Hébert et Marleau (1991)). Ces valeurs moyennes peuvent être obtenues

par interpolations directes (en fonction du paramètre de dilution calculé par le modèle d'autoprotection) dans des tables provenant du traitement des sections efficaces continues en énergie par un logiciel tel que NJOY99 (Macfarlane et Muir (2000)). Les sections efficaces moyennes ainsi obtenues sont ensuite multipliées par des facteurs d'équivalence qui permettent de prendre en compte les effets non-linéaires de condensation. L'objectif de ce calcul est donc de calculer la section efficace microscopique auto-protégée $\overline{\sigma}_{y,g}$ de l'isotope y dans le groupe énergétique g qui est définie ainsi (u étant la léthargie) :

$$\overline{\sigma}_{y,g} = \mu_g \frac{\int_{u_{g-1}}^{u_g} \sigma_y(u)\phi(u)}{\int_{u_{g-1}}^{u_g} \phi(u)} \tag{2.39}$$

où μ_g est un facteur de superhomogénéisation (SPH) obtenu par une procedure d'équivalence multigroupe. La difficulté de cette étape réside dans le fait qu'à ce stade du calcul, nous n'avons pas encore évalué le flux (puisqu'il nécessite de connaître les sections efficaces auto-protégée). Il faudra donc réaliser certaines approximations pour obtenir un estimé du flux qui nous permette de calculer les sections efficaces auto-protégée par l'équation (2.39). Comme nous le verrons par la suite, la façon de réaliser le calcul d'auto-protection des résonances est un paramètre très sensible pour les résultats finaux. Dans ce travail, nous utiliserons la méthode de Stamm'ler généralisée qui a été implantée dans DRAGON (Hébert et Marleau (1991)).

2.8 Les modèles de fuite

Un modèle de fuite, ou modèle de *mode fondamental*, est indispensable pour traiter les cellules ou les assemblages qui se retrouveront dans un réacteur fini, en particulier s'ils sont étudiés avec un modèle en deux dimensions ou s'ils sont entourés de conditions frontières de réflexion. On utilise généralement un modèle de fuite B_n (Petrovic et Benoist (1996)) pour simuler les fuites de neutrons dans la cellule ou l'assemblage.

Dans un calcul de réseau, nous ne savons pas *a priori* dans quel état se trouve la cellule ou l'assemblage à l'intérieur du cœur, la meilleure approximation est donc de considérer que son état est stationnaire, c'est-à-dire que $k_{\text{eff}} = 1$. On adopte la stratégie suivante :

- Le calcul de flux à l'intérieur de la cellule ou de l'assemblage va être effectué dans des conditions sans fuite.
- Un modèle de fuite va être utilisé pour forcer $k_{\text{eff}} = 1$. En général, on utilise pour ce faire l'approximation du mode fondamental qui consiste à représenter le flux comme le produit d'une distribution macroscopique dans l'espace $\psi(\vec{r})$ et d'un flux fondamental

périodique ou homogène $\varphi(\vec{r}, E, \widehat{\Omega})$.

$$\Phi(\vec{r}, E, \widehat{\Omega}) = \psi(\vec{r})\varphi(\vec{r}, E, \widehat{\Omega}) \tag{2.40}$$

– Dans le cas d'un réseau périodique de cellules ou d'assemblages, $\psi(\vec{r})$ est supposée être une propriété du cœur au complet et est solution de l'équation de Laplace :

$$\nabla^2\psi(\vec{r}) + B^2\psi(\vec{r}) = 0 \tag{2.41}$$

où le laplacien B^2 est un nombre réel qui sert à ajuster $\psi(\vec{r})$ de telle manière que $k_{\text{eff}} = 1$.

Sans la connaissance du flux dans le réacteur au complet, nous utilisons la solution générique de l'équation (2.41) :

$$\psi(\vec{r}) = \psi_0 e^{i\overrightarrow{B}\cdot\overrightarrow{r}} \tag{2.42}$$

où le vecteur $\overrightarrow{B}$ est choisi de telle manière que son module vérifie l'équation (2.41). Nous allons supposer que nous pouvons calculer le taux de fuite en considérant que la cellule est homogène (le calcul des autres grandeurs se fait toujours avec une cellule hétérogène). Le flux de neutrons s'écrit alors : $\phi(\vec{r}, E, \widehat{\Omega}) = \varphi(E, \widehat{\Omega})e^{i\overrightarrow{B}\cdot\overrightarrow{r}}$ où $\varphi(\vec{r}, E, \widehat{\Omega})$ est une grandeur complexe. On peut alors écrire l'équation (2.12) de la manière suivante :

$$\begin{aligned}\left[\Sigma(E) + i\overrightarrow{B}\cdot\widehat{\Omega}\right]\varphi(E, \widehat{\Omega}) = &\int_{4\pi} d^2\Omega' \int_0^\infty dE'\Sigma_s(E \leftarrow E', \widehat{\Omega} \leftarrow \widehat{\Omega}')\varphi(E), \widehat{\Omega}') \\ &+ \frac{\chi(E)}{4\pi k_{\text{eff}}}\int_0^\infty dE'\nu\Sigma_f(E')\varphi(E')\end{aligned} \tag{2.43}$$

On développe la section efficace de diffusion à l'ordre 1 en polynôme de Legendre :

$$\Sigma_s(E \leftarrow E', \widehat{\Omega} \leftarrow \widehat{\Omega}') = \frac{1}{4\pi}\Sigma_{s,0}(E \leftarrow E') + \frac{3}{4\pi}\Sigma_{s,1}(E \leftarrow E')\widehat{\Omega}\cdot\widehat{\Omega}' \tag{2.44}$$

On intègre alors l'équation (2.43) avec deux pondérations différentes. On obtient au final l'équation suivante :

$$\begin{aligned}\left[\Sigma(E) + d(B, E)B^2\right]\varphi(E) = &\int_0^\infty dE'\Sigma_{s,0}(E \leftarrow E')\varphi(E') \\ &+ \frac{\chi(E)}{4\pi k_{\text{eff}}}\int_0^\infty dE'\nu\Sigma_f(E')\varphi(E')\end{aligned} \tag{2.45}$$

avec $d(B, E) = \frac{i}{B^2\varphi(E)}\overrightarrow{B}\cdot\int_{4\pi} d^2\Omega\widehat{\Omega}\varphi(E, \widehat{\Omega})$ le coefficient de fuite qui dépend notamment de

$\Sigma_{s,1}(E \leftarrow E')$. Il est aisé de passer de ces valeurs à des grandeurs multigroupes.

Nous pouvons utiliser trois hypothèses distinctes pour réaliser le calcul de $d(B, E)$ (Petrovic et Benoist (1996)) :

- L'hypothèse B0 homogène : On suppose que la section efficace de diffusion est isotrope ($\Sigma_{s,1}(E \leftarrow E') = 0$) et que $d(B, E)$ ne dépend pas de la région.
- L'hypothèse B1 homogène : On suppose que la section efficace de diffusion est anisotrope ($\Sigma_{s,1}(E \leftarrow E') \neq 0$) et que $d(B, E)$ ne dépend pas de la région.
- L'hypothèse B1 hétérogène : On suppose que la section efficace de diffusion est anisotrope ($\Sigma_{s,1}(E \leftarrow E') \neq 0$) et que $d(B, E)$ dépend de la région.

Dans le cadre de la méthode des probabilités de collision, le modèle de fuite peut être inséré de différentes manières. L'une d'elle consiste à remplacer l'équation (2.33) par l'équation :

$$\mathbf{\Phi}_g = \mathcal{W}_g[\mathbf{Q}_g^* - d_g(B)B^2\mathbf{\Phi}_g] \tag{2.46}$$

2.9 Homogénéisation et condensation

À l'issue d'un calcul de flux à partir de l'équation du transport dans le formalisme multigroupe, nous obtenons des flux, des taux de réaction et des sections efficaces pour chaque région et chaque groupe d'énergie. Or pour l'étape suivante du schéma de calcul, nous avons souvent besoin de propriétés moyennées sur des macro-régions (homogénéisation) et des macro-groupes d'énergie (condensation).

Les grandeurs sont définies sur un espace à $N \times G$ dimensions (N régions et G groupes d'énergie). On cherche de nouvelles grandeurs sur un espace à $M \times K$ dimensions de telle manière que :

- à chaque indice $k \in [1, K]$ correspond un ensemble d'indices $g \in [1, G]$: G_k
- à chaque indice $m \in [1, M]$ correspond un ensemble d'indices $n \in [1, N]$: N_m

On a évidemment : $G = \bigcup_{k=1}^{K} G_k$ et $N = \bigcup_{m=1}^{M} N_m$

Les propriétés homogénéisées et condensées sont définies de manière à conserver les flux

et les taux de réaction. On applique les formules suivantes :

$$V_m = \sum_{i \in N_m} V_i \tag{2.47}$$

$$\phi_m^k = \frac{1}{V_m} \sum_{i \in N_m} \sum_{g \in G_k} V_i \phi_i^g \tag{2.48}$$

$$\Sigma_m^k = \frac{1}{V_m \phi_{m,k}} \sum_{i \in N_m} \sum_{g \in G_k} V_i \phi_i^g \Sigma_i^g \tag{2.49}$$

$$\Sigma_{s,m}^{k \leftarrow l} = \frac{1}{V_m \phi_m^l} \sum_{i \in N_m} \sum_{g \in G_k} \sum_{h \in G_l} V_i \phi_i^h \Sigma_{s,i}^{g \leftarrow h} \tag{2.50}$$

$$\nu\Sigma_{f,m}^k = \frac{1}{V_m \phi_m^k} \sum_{i \in N_m} \sum_{g \in G_k} V_i \phi_i^g \nu\Sigma_{f,i}^g \tag{2.51}$$

$$\chi_m^k = \frac{\sum_{i \in N_m} V_i \sum_{j=1}^{J^{\text{fiss}}} \sum_{g \in G_k} \chi_j^g \sum_{g'=1}^{G} \phi_i^{g'} \nu\Sigma_{f,i,j}^{g'}}{V_m \sum_{j=1}^{J\text{fiss}} \sum_{h=1}^{K} \nu\Sigma_{f,m,j}^h \phi_m^h} \tag{2.52}$$

Dans l'ensemble de ce travail, on considérera les sections efficaces calculées lors du calcul de réseau en deux macro-groupes énergétiques séparés à 0,625 eV. Le groupe le plus énergétique sera appelé rapide et le moins énergétique sera appelé thermique.

2.10 Équivalence transport-transport transport-diffusion

Si nous réalisons un nouveau calcul de transport avec les sections efficaces homogénéisées et condensées, nous n'obtiendrons pas exactement les mêmes taux de réaction. Ceci est dû au fait que l'équation de transport n'est pas linéaire par rapport aux sections efficaces. Pour garantir l'égalité entre les anciens et les nouveaux taux de réaction, on modifie les sections efficaces homogénéisées et condensées de toutes les réactions en les multipliant par un unique coefficient $\mu_{m,k}$ appelé facteur d'équivalence. Cette étape est appelée l'équivalence transport-transport (Hébert et G. (1993)).

On utilise la même technique pour réaliser une équivalence entre le calcul de transport et le calcul de diffusion. Dans le cas d'une équivalence transport-diffusion, il faut en plus corriger certains effets d'approximation de la diffusion. En particulier, on peut utiliser la normalisation de Selengut (Selengut (1960)) qui consiste à introduire des facteurs de discontinuité qui induisent une condition de discontinuité aux interfaces entre assemblages dans le calcul de cœur. Il ssont utilisés dans les calculs de diffusion homogène en normalisant simplement les sections effiaces et les coefficients de diffusion.

Dans ce travail, nous n'avons pas fait d'études sur l'influence des différentes techniques d'équivalence disponibles dans DRAGON.

2.11 Calcul d'évolution

Le calcul d'évolution consiste à suivre les concentrations isotopiques des mélanges au cours du temps. L'objectif est de déterminer le bilan matière en fonction du burnup.

Les équations qui régissent les concentrations isotopiques sont les équations de Bateman :

$$\frac{dN_i}{dt} = \sum_j [(\sigma_{j\to i} + Y_i\sigma_{f,j})\phi + \lambda_{j\to i}]N_j - (\sigma_{a,i} + \lambda_i)N_i \tag{2.53}$$

où

- N_i est la concentration de l'isotope i
- $\sigma_{j\to i}$ est la section efficace des réactions conduisant à la foramtion de l'isotope i à partir de l 'isotope j
- Y_i est le rendement de fission : c'est la probabilité qu'à l'issue d'une fission l'isotope i soit produit (elle est nulle pour un atome lourd)
- $\sigma_{f,j}$ est la section de fission de l'isotope j
- $\lambda_{j\to i}$ est la constante de décroissance radioactive de l'isotope j qui conduit à la formation de l'isopote i
- $\sigma_{a,i}$ est la section d'absorption de l'isotope i
- λ_i est la constante de décroissance radioactive de l'isotope i

Ces équations forment un système d'équations différentielles couplées qui est résolu par un processus itératif. Celui-ci repose sur l'hypothèse que le flux neutronique est constant sur une certaine période de temps. Il est donc nécessaire de découper le temps en intervalles (pas de burnup) où cette hypothèse est vérifiée.

En résolvant ce système pour les différents pas de burnup, on peut ainsi construire une bibliothèque de sections efficaces qui sea utilisée par le code de cœur qui réalisera l'interpolation des sections effiaces à la valeur de burnup requise.

2.12 Équation de la diffusion

Le calcul complet du flux neutronique dans le cœur d'un réacteur de puissance est pour l'heure trop complexe pour être mené avec l'équation de transport. L'opérateur angulaire *toto* est la source de beacoup de difficultés de traitement comme nous avons pu le montrer précédemment. L'objectif que nous devons nous fixer est donc de réaliser des approximations sur le flux neutronique qui permettenet de simplifier cette expression.

2.12.1 Loi de Fick

Approximation P_1

Pour résoude l'équation de transport que nous avons présentée au début de ce chapitre, on effectue un développement du flux neutronique en hamroniques sphériques en conservant les $N+1$ premiers termes (Bell et Glasstone (1970)). C'est ce qu'on appelle l'approximation P_1 où seuls les deux premiers termes sont conservés. Le flux s'écrit donc :

$$\phi^g(\vec{r},\widehat{\Omega}) \approx \frac{1}{4\pi}[\phi^g(\vec{r}) + 3\widehat{\Omega}\cdot\overrightarrow{J}_g(\vec{r})] \tag{2.54}$$

où $\overrightarrow{J}_g(\vec{r})$ est la densité de courant angulaire : $\overrightarrow{J}_g(\vec{r}) = \int_{4\pi} d^2\Omega\widehat{\Omega}\cdot\phi^g(\vec{r},\widehat{\Omega})$.

De même, on développe à l'ordre 1 en série harmonique la section efficace de diffusion. On trouve :

$$\Sigma_s^g(\vec{r},\widehat{\Omega}\cdot\widehat{\Omega}') \approx \frac{1}{4\pi}\left[\Sigma_{s,0}^g(\vec{r}) + 3\widehat{\Omega}\cdot\widehat{\Omega}'\Sigma_{s,1}^g(\vec{r})\right] \tag{2.55}$$

En remplaçant l'équation (2.20) et en intégrant sur l'angle solide, on obtient alors le bilan de population neutronique simplifié :

$$\begin{aligned}\Sigma^g(\vec{r})\phi^g(\vec{r},\widehat{\Omega}) + \vec{\nabla}\cdot\vec{J}(\vec{r}) = &\sum_{g'=0}^{G}\Sigma_{s,0}^{g\leftarrow g'}\phi^{g'}(\vec{r}) \\ &+ \frac{\chi(E)}{k_{\text{eff}}}\sum_{g'=0}^{G}\nu\Sigma_f^g(\vec{r})\phi^{g'}(\vec{r})\end{aligned} \tag{2.56}$$

Utilisation de la loi de Fick

L'équation précédente dépend de deux variables : le flux et le courant neutroniques. On utilise alors la loi de Fick :

$$\vec{J^g}(\vec{r}) = -D^g(\vec{r})\vec{\nabla}\phi^g(\vec{r}) \tag{2.57}$$

On utilise la même forme de flulx que pour le calcul de fuite de type B_n : $\phi^g(\vec{r}) = \varphi^g e^{\vec{B}\cdot\vec{r}}$. On écrit également : $\vec{J^g}(\vec{r}) = \vec{\mathcal{J}}^g e^{\vec{B}\cdot\vec{r}}$. En remplaçant dans l'équation (2.57), on trouve :

$$D_g = \frac{1}{B}\frac{i\mathcal{J}^g}{\varphi^g} \tag{2.58}$$

En remplaçant dans l'équation (2.56) on trouve :

$$-\vec{\nabla}\cdot D_g(\vec{r})\vec{\nabla}\phi^g(\vec{r}) + \Sigma_0^g(\vec{r})\phi^g(\vec{r}) = \sum_{g'=0}^{G}\Sigma_{s,0}^{g\leftarrow g'}\phi^{g'}(\vec{r}) + \frac{1}{k_{\text{eff}}}\chi^g(\vec{r})\sum_{g'=0}^{G}\nu\Sigma_f^g(\vec{r})\phi^{g'}(\vec{r}) \tag{2.59}$$

cette équation peut se mettre sous la forme matricielle suivante :

$$\left(\lambda[\chi][\nu\Sigma_f]^T - ([\Sigma] - \vec{\nabla}\cdot[D]\vec{\nabla})\right)[\phi] = 0 \tag{2.60}$$

avec $\lambda = \frac{1}{k_{\text{eff}}}$. Ègauche dans la parenthèse, le terme de production. À droite le terme d'élimination. Ceci est un problème aux valeurs propores. Le flux nul est une solution triviale de cette équation. Il existe de nombreuses solutions non triviales pour différentes valeurs propres de λ. Celle qui correspond à la plus grande des valuers absolues de k_{eff} est le mode fondamental et nous donne le k_{eff} du réacteur. Seule cette solution a une signification physique car c'est la seule qui conduit à des flux de neutrons entièrement positifs. Les autres vecteurs propres sont appelés des harmoniques et peuvent être utiles pour des calculs de perturbation. k_{eff} correspond physiquement au rapport entre les nombres de neutrons entre deux générations successives. Ainsi, ondira qu'un réacteur est critique si $k_{\text{eff}} = 1$, sous-critique si $k_{\text{eff}} < 1$ et sur-critique si $k_{\text{eff}} > 1$.

À l'état stationnaire, un réacteur doit être critique. Il importe donc de contrôler précisemment la réactivité de l'ensemble. Cela n'empêche pas certaines zones d'être sur-critiques, mais cela impose qu'elles soient compensées par des régions sous-critiques.

2.12.2 Conditions aux frontières

De même que pour l'équation de transport, l'équation de diffusion doit être complétée par des conditions frontières pour fermer le système. Le flux de neutrons est une distribution continue et le courant de neutrons doit être continu à travers n'importe qu'elle surface virtuelle.

Soit un plan d'abscisse x_0 et de normale unitaire $\overrightarrow{N}$, les deux conditions de continuité s'écrivents :

$$\forall y \text{ et } z \qquad \phi^g(x_0^-, y, z) = \phi^g(x_0^+, y, z) \tag{2.61}$$

$$\forall y \text{ et } z \qquad \vec{J}_g(x_0^-, y, z)\cdot\vec{N} = \vec{J}_g x_0^+, y, z)\cdot\vec{N} \tag{2.62}$$

En utilisant la loi de Fick, cette dernière équation peut s'écrire : $\forall y et z$

$$-D_g(x_0^-, y, z)\frac{d}{dx}\phi^g(x, y, z)|_{x=x_0^-} = -D_g(x_0^+, y, z)\frac{d}{dx}\phi^g(x, y, z)|_{x=x_0^+} \tag{2.63}$$

Cette équation fait apparaître clairement que le gradient du flux est discontinu chaque fois que le coefficient de diffusion est discontinu, i.e. à chaque changement de milieu. Ces conditions frontières doivent être vérifiées sur l'ensemble de la limité où nous calculons le flux (donc les limites du réacteur), sauf si certaines symétries nous permettent de réduire notre domaine d'étude.

2.12.3 Discrétisation de l'équation de diffusion

La discrétisation de l'équation de diffusion consiste à transformer l'équation différen-tielle en un système matriciel qui peut-être résolu par des techniques d'analyse numérique. Il existe une grande variété de méthodes de discrétisation de l'équation de diffusion ; dans ce projet nous n'en présenterons qu'une seule : la méthode des différences finies centrées.

2.13 Présentation des codes de calculs

2.13.1 Le code d'assemblage DRAGON4

Le code DRAGON4 (Marleau *et al.* (2010)) créé par l'Institut du Génie Nucléaire de l'École Polytechnique de Montréal (IGN) résulte d'un effort de rationalisation concerté qui consistait à unifier en un seul code differents modèles et algorithmes utilisés pour la résolution de l'équation de transport des neutrons. Ainsi le code de cellule DRAGON est divisé en différents modules de calculs qui sont reliés entre eux via le programme de contrôle généralisé du groupe d'analyse nucléaire GANLIB (Hébert et Roy (1994)), l'échange d'information entre les différents modules etant assuré par des structures de données. Le langage de programmation est le FORTRAN77.

Dans ce projet nous travaillerons avec la version 4 du logiciel. Les principales composantes que nous utiliserons sont :

– Les modules d'analyse de géométrie générant des fichiers de lignes d'intégration appelés "tracking". Nous utiliserons dans nos études trois d'entre eux : le module SYBILT: qui crée des ligne d'intégration pour un calcul aux courants d'interfaces en milieu hétérogène, le module NXT: qui crée des lignes d'intégration pour un calcul avec la méthode des CP et le module MCCGT: qui vient par dessus le module NXT: pour ajouter la possibilité d'effectuer un calcul avec la méthode des caractérisiques. DRAGON permet deux types de conditions frontières de réflexion pour les lignes d'intégration (pour les

modules MCCGT: et NXT:) : la condition de réflexion spéculaire et la condition de réflexion blanche ou isotrope. DRAGON peut utiliser des géométries 1-D, 2-D et 3-D. Dans ce travail, nous n'utiliserons que des modèles en deux dimensions.

- Le module de calcul des sections efficaces auto-protégées USS : utilisant la méthode des sous-groupes.
- Le module de calcul des probabilités de collision à partir des fichiers de lignes d'intégration : ASM: .
- Le module de résolution pour le flux multigroupe utilisant les probabilités de collision : FLU: .
- Le module d'édition EDI: qui réalise l'homogénéisation et la condensation des sections efficaces.

2.13.2 Le code de cœur DONJON4

Le code informatique DONJON4 est en développement permanent. Il dépend de l'exécution d'autres codes composant de la Version4 (Hébert (2006)), notamment : GANLIB, UTILIB, DRAGON (Hébert *et al.* (1995)), et TRIVAC (Hébert (1987b)). Les modules de DRAGON sont utilisés au sein de DONJON pour définir la géométrie, fournir les bibliothèques de sections efficaces macroscopiques et interpoler les résulats du code d'assemblage. Le solveur TRIVAC est utilisé pour effectuer une discrétisation spatiale de la géométrie du réacteur et fournir une solution numérique selon les choix de l'utilisateur. Comme l'ensemble de la distribution Version4, son approche est modulaire (cf. figure 2.1). Le code UTILIB fournit des outils ainsi que la bibliothèque d'algèbre linéaire. Pour finir, GANLIB s'assure de l'exécution cohérente de l'ensemble de ces codes. L'unique langage de programmation est le FORTRAN77.

DONJON4 est découpé en modules, chacun effectuant une tâche spécifique. Ils ont été conçus en premier lieu pour modéliser un cœur complet de réacteur en géométrie 3D cartésienne. Ces modules ont conçus pour modéliser au mieux des assemblages typiques de réacteurs CANDU, mais peuvent aussi être utilisé pour modéliser des réacteur REP. Le code DONJON4 peut effecteur de nombreuses analyses statiques sur un cœur entier et être utilisé pour en déterminer les caractéristiques clefs, à savoir la puissance et la distribution de flux normalisé sur l'ensemble du réacteur.

Chaque assemblage dans DONJON4 est composé d'un nombre fixé de "bundles", chacun ayant ces propriétés qui sont récupérées et interpolées selon des paramètres globaux et locaux à partir d'une bibliothèque macroscopique. Celle-ci s'obtient avec le code DRAGON. Deux bibliothèques distinctes doivent être construites avec DONJON : une première pour les matériau dont les propriétés sont indépendantes de l'évolution du combustible (comme

le réflecteur), et une deuxième uniquement pour les matériaux combustibles. Les deux bibliohtèques sont alors fusionnées pour en produire une utilisable par TRIVAC pour obtenir une solution numérique.

Les modules suivants de DRAGON peuvent être utilisés dans DONJON :

- le module GEO: crée ou modifie une géométrie de réacteur. La disposition spatiale des mixtures doit être aussi définie avec ce module. Seules des géométries 3D cartésiennes ou hexagonales avec éléments homogènes et mesh-splitting uniforme (i.e. régulières) sont autorisées dans DONJON.
- le module MAC: crée ou modifie une *macrolib* contenant les propriétés d'un matériau en spécifiant "à la main" les sections macroscopiques pour chaque groupe d'énergie.
- le module NCR: crée une *macrolib* contenant les propriétés des matériaux en interpolant le données nucléaires d'une bibliothèque générée par DRAGON4

Les modules suivants de TRIVAC peuvent être utilisés dans DONJON :

- le module TRIVAT: effectue une discrétisation numérique 3D ou "tracking" de la géométrie du réacteur. L'utilisateur peut choisir suivant les discrétisations suivantes : éléments finis et différences finies centrées d'ordre 1, 2 et 3.
- le module TRIVAA: calcule le système de matrice à l'aide de l'objet "tracking" précédemment créé.
- le module FLUD: calcule la solution numérique aux valeurs propres du système de matrice précédemment obtenu. Peut utilisé des méthodes de éléments finis ou de différences finis.

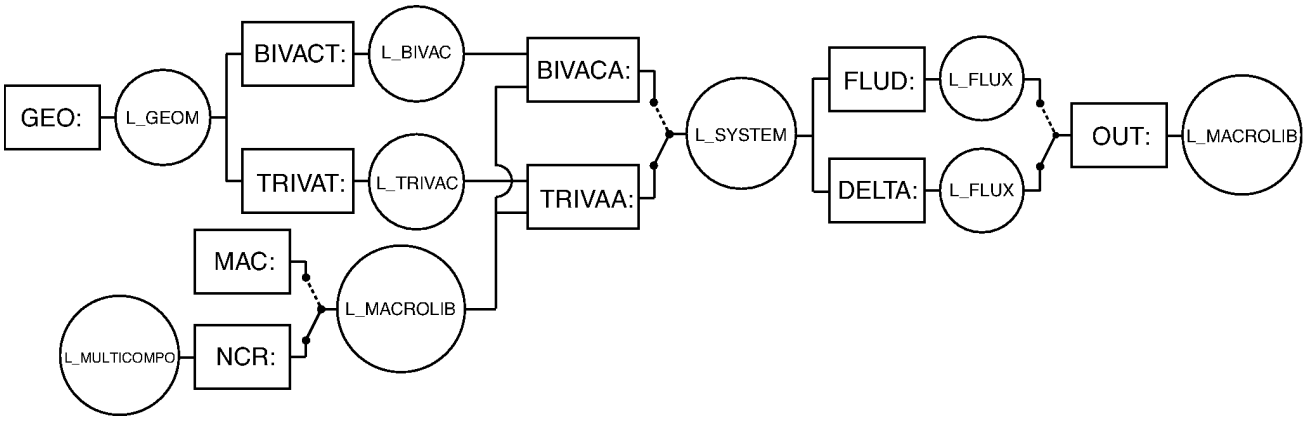


Figure 2.1 Approche modulaire de TRIVAC

2.13.3 Le code de cœur COCAGNE

Le code informatique COCAGNE est développé par EDF à Clamart pour une utilisation pour l'instant réservée à la recherche et au développement. Le langage de programmation est le C++, mais son interface utilisateur est une sur-couche Python (van Rossum (1995)).

Méthodes de résolution, conditions aux limites, langage.

Il utilise des bibliothèques *DKLib* qui stockent séparemment les sections macroscopiques et les facteurs d'équivalence normalisés (comme les facteur de Selengut), contrairement aux *macrolib* de DRAGON4 qui stockent les sections modifiées par le facteur SPH. Il est donc possible au sein de COCAGNE de décider ou non d'appliquer les facteurs d'équivalence ou de calculer de nouveaux facteurs d'équivalence.

CHAPITRE 3

CAMPAGNE SIMPLIFIÉE DONJON4 VS COCAGNE

L'objectif de cette partie est d'effectuer deux fois le même calcul de longueur de campagne d'un REP, l'un avec le code DONJON4 et l'autre avec le code COCAGNE afin d'assurer leur cohérence. Lors de ce test, les sections efficaces requises par le calcul de cœur seront interpolées dans la MULTICOMPO ou la DKLIB (Hébert (2009)). Le nouveau calcul d'historique ne sera pas utilisé.

Dans un premier temps, nous comparerons simplement les deux bibliothèques de sections efficaces produites par DRAGON4 : une pour DONJON4 (MULTICOMPO) et une pour COCAGNE (DKLIB). Ensuite nous comparerons les grandeurs physiques en sortie des codes de cœur sur des géométries simplifiées. Puis nous complexifierons de plus en plus la géométrie jusqu'à atteindre le cœur du REP de la longueur de campagne.

Une fois les comparaisons validées en mode statique, une évolution du combustible sera simulée dans les codes de cœur avec le module TINST: . Pour finir un calcul sera effectué sur toute la longueur de campagne.

Cette partie se conclura sur une réflexion à propos de la modélisation des mécanismes de contre-réaction dans les codes de cœur DONJON4 et COCAGNE.

3.1 Le calcul d'assemblage

Le calcul d'assemblage est un prérequis du calcul de cœur. Il est effectué par DRAGON4, et est identique dans les deux cas. Il consiste à calculer les propriétés d'un assemblage typique, élément constitutif du cœur.

Partant d'une bibliothèque de sections microscopiques à 281 groupes d'énergie, notre but est d'obtenir une bibliothèque de sections macroscopiques à 2 groupes d'énergie paramétrée en température combustible, température et densité modérateur, partie pour million (ppm) de bore, niveau de xénon et burnup.

La bibliothèque macroscopique contiendra 3 valeurs de température combustible, 4 valeurs de température et densité modérateur, 4 valeurs de bore, 2 valeurs de xénon (saturé et nul) et 28 valeurs de burnup. Au total elle aura 3360 tuples, chaque tuple correspondant à un calcul élémentaire DRAGON4. Il est entendu qu'un calcul élémentaire DRAGON correspond à une seule étape tabulée en burnup.

La création de cette bibliothèque macroscopique se fait avec un calcul de fil et plusieurs calculs de reprise. Le calcul de fil sert à connaître les concentrations isotopiques aux 28 valeurs de burnup ciblées (ayant fixé les autres paramètres à leur valeur nominale). Un calcul de reprise consiste à alimenter la bibliothèque macroscopique et nécessite les concentrations isotopiques déterminées par le calcul de fil.

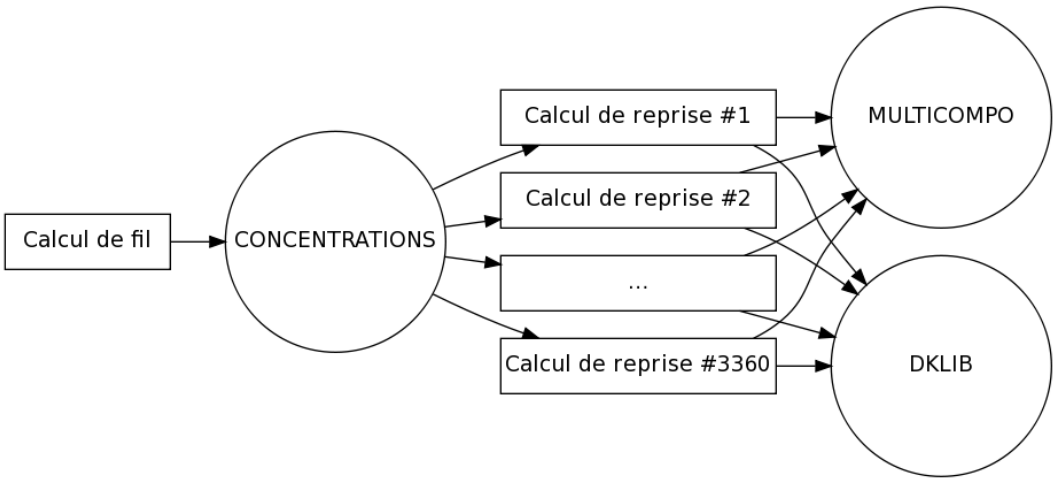


Figure 3.1 Calcul d'assemblage

3.1.1 La géométrie de l'assemblage

L'assemblage est composé 17×17 cellules, dont 259 sont des crayons combustibles et 30 sont des trous d'eau (voir figure 1.2).

Il est modélisé suivant deux niveaux de précision que l'on appelera géométrie N1 (géométrie niveau 1) et géométrie N2 (géométrie niveau 2), la géométrie N2 étant plus fine que la géométrie N1. Pour des raisons de symétrie, il est possible de ne modéliser qu'un huitième de l'assemblage.

Dans les géométries N1 et N2 (figure 3.2), le côté d'une cellule est d'environ 1,25 centimètres. Au centre de cette cellule, un crayon est représenté par un cylindre enrobé de plusieurs gaines. À l'intérieur on trouve de l'uranium enrichi à 3,7%, puis une première zone de vide (permettant d'encaisser la dilatation de la pastille et d'accueillir les gaz issus de l'irradiation), elle-même entourée par une gaine d'alliage de zirconium, le tout plongé dans de l'eau contenant du bore dilué.

Dans la géométrie N2 (figure 3.3), des raffinements sont apportés par rapport à la géométrie N1. D'une part les cellules ont un découpage supplémentaire (cartésien 3×3), et d'autre part une lame d'eau est ajoutée sur les côtés extérieurs de l'assemblage.

Une troisième géométrie est nécessaire pour effectuer l'autoprotection (figure 3.4). Elle est semblable à la géométrie N1, la seule différence étant la gaine extérieure remplie d'eau de la géométrie N1 qui est absente dans la géométrie d'autoprotection, ce qui est normal car c'est une géométrie simplifié avec des regroupements de cellule.

3.1.2 Le calcul de fil

Le but du calcul de fil est de connaître les concentrations isotopiques aux 28 valeurs de burnup que l'on cherche à obtenir dans la bibliothèque macroscopique.

On construit une bibliothèque microscopique à valeurs nominales, et on convertit notamment la concentration de bore de ppm en g.cm^{-3}. Cette conversion fait intervenir la densité de l'eau, si bien que ces deux paramètres semblent a priori couplé. Nous verrons dans la section 3.1.3 comment faire pour que ce ne soit pas le cas.

Le schéma de calcul retenu est un schéma à deux niveaux (Reysset (2009); Vallerent (2009)), qui nécessite trois géométries : la géométrie N1 (figure 3.2), la géométrie N2 (figure 3.3) et la géométrie d'autoprotection (figure 3.4) :

- L'équation du transport, sur la géométrie N1 contenant 40 mixtures définies pour 281 groupes d'énergie, est résolue par un calcul de type B avec des conditions aux limites de réflexion (milieu infini). L'inventaire isotopique de ces 40 mixtures est initialisé à burnup nul.

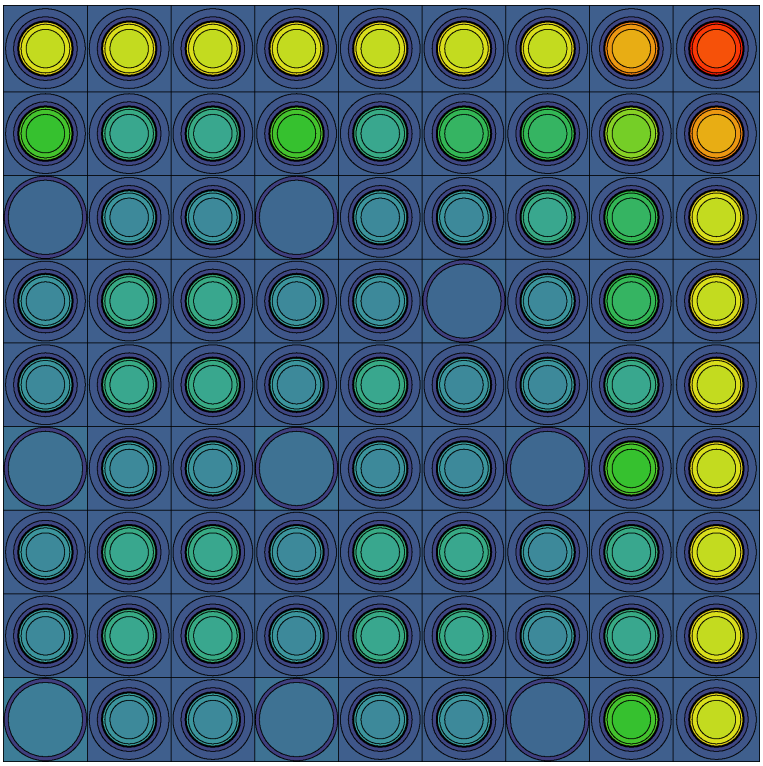


Figure 3.2 Géométrie niveau 1, quart nord-est

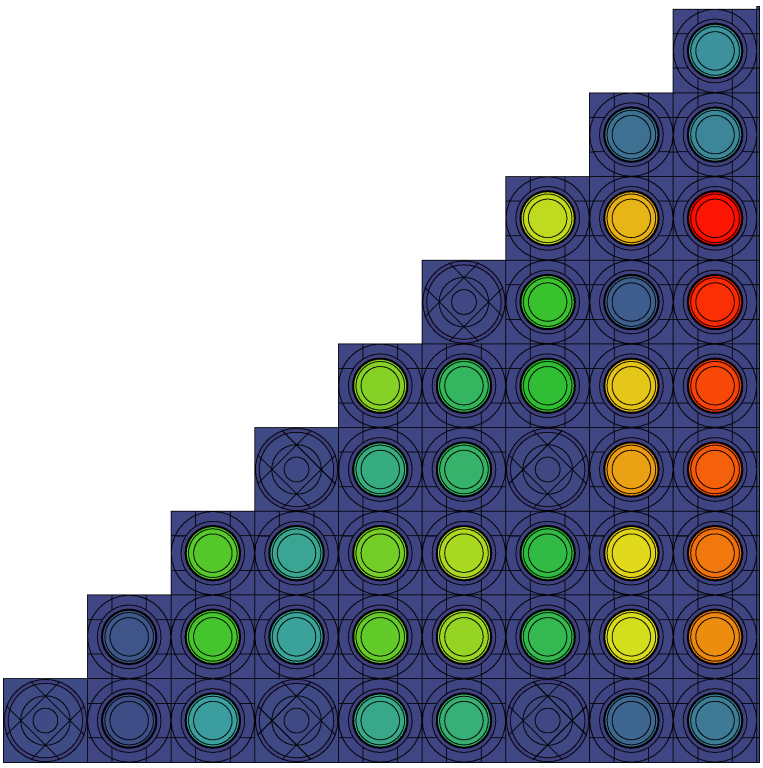
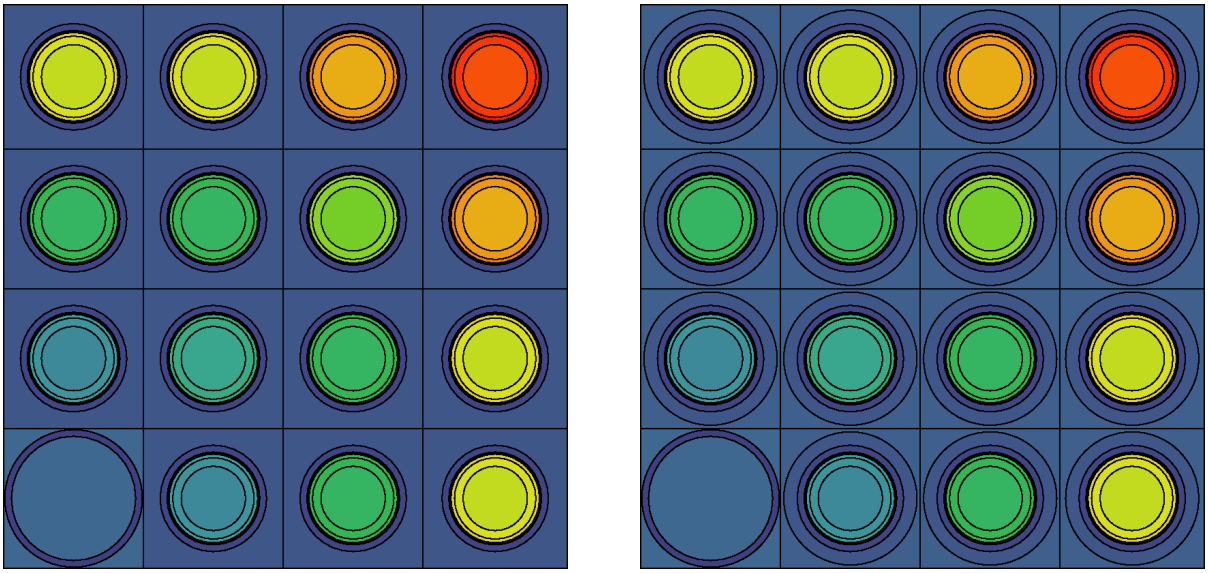


Figure 3.3 Géométrie niveau 2, huitième est-nord-est



(a) Géométrie d'auto-protection (b) Géométrie niveau 1

Figure 3.4 Différences entre les géométries d'auto-protection et niveau 1

- Le flux N1 issu du calcul précédent est utilisé pour condenser les mixtures à 26 groupes d'énergie, que l'on duplique à hauteur de 164 mixtures (stretching pour fitter la géométrie N2).
- L'équation du transport, sur la géométrie N2 contenant 164 mixtures définies pour 26 groupes d'énergie, est résolue par un calcul de type K avec des conditions aux limites de réflexions (milieu infini).
- Le flux N1 issu du calcul précédent est utilisé pour faire évoluer les mixtures au burnup suivant.
- Les 164 mixtures sont alors condensées à 1 groupe et refusionnées pour reformer les 40 mixtures initiales, et ainsi obtenir les concentrations isotopiques recherchées.

Ce processus est répété 73 fois car pour obtenir ces 28 concentrations, il est nécessaire de passer par des points de burnup intermédiaires afin de conserver une bonne finesse dans l'évolution du combustible.

3.1.3 Les calculs de reprise

Le but des calculs de reprise est de créer la bibliothèque de sections efficaces macroscopiques paramétrée en température combustible, température et densité modérateur, partie pour million (ppm) de bore, niveau de xénon et burnup.

Le calcul de reprise consiste à modifier les conditions de calculs pour un inventaire isotopique donné (une bibliothèque microscopique) issu du calcul de fil pour ensuite reprendre le calcul de fil (section 3.1.2). Les modifications consistent à changer la température du combustible, la température et la densité de l'eau ainsi que les concentrations de xénon et de bore. Ces changements sont tous triviaux à l'exception de la concentration en bore. En effet celle-ci est toujours calculée par rapport à la densité d'eau nominale, que ce soit ou non le cas. Ce choix se justifie d'une part par le fait que la densité d'eau varie peu et n'influe pas de façon significative sur la conversion de ppm à $g.cm^{-3}$ pour le bore et d'autre part car on ne souhaite pas avoir de couplage entre les paramètres densité d'eau et concentration en bore. Enfin, comme l'unité de stockage est le $g.cm^{-3}$, on aura bien 4 valeurs pour le bore, une par concentration en ppm, ce qui n'aurait pas été le cas si on avait convertit les ppm en se servant de différentes densités d'eau.

Le schéma de calcul retenu est le même schéma à deux niveaux (Reysset (2009); Vallerent (2009)) que le calcul de fil :

- L'équation du transport, sur la géométrie N1 contenant 40 mixtures définies pour 281 groupes d'énergie, est résolue par un calcul de type B avec des conditions aux limites de réflexion (milieu infini). L'inventaire isotopique de ces 40 mixtures est initialisé soit à burnup nul, soit à burnup non nul à partir des résultats du calcul de fil.

- Le flux N1 issu du calcul précédent est utilisé pour condenser les mixtures à 26 groupes d'énergie, que l'on duplique à hauteur de 164 mixtures (stretching pour fitter la géométrie N2).
- L'équation du transport, sur la géométrie N2 contenant 164 mixtures définies pour 26 groupes d'énergie, est résolue par un calcul de type K avec des conditions aux limites de réflexions (milieu infini).
- Le flux N1 issu du calcul précédent est utilisé pour condenser les 164 mixtures à 2 groupe d'énergie et calculer les facteur d'équivalence de Selengut.
- Les 164 mixtures sont alors fusionnées en une seule (dont les isotopes sont fusionnés en un seul isotope macroscopique résiduel) et celle-ci est agrégée au bon endroit dans la bibliothèque de sections efficaces macroscopiques.

Pour un calcul de reprise, ce processus est répété 28 fois : une fois pour chaque point de burnup de la bibliothèque macroscopique. Dans le cas qui nous intéresse, il y a 120 calculs de reprises ($3360/28 = 120$).

La dernière étape est différente si l'on créé une DKLIB plutôt qu'une MULTICOMPO. La DKLIB nécessite de particulariser certains isotopes (U^{234}, U^{235}, U^{236}, U^{237}, U^{238}, Np^{237}, Np^{238}, Np^{239}, Pu^{238}, Pu^{239}, Pu^{240}, Pu^{241}, Pu^{242}, Cm^{242}, Cm^{243}, Cm^{244}, Cm^{245}, Am^{241}, Am^{242M}, Am^{243}, Pm^{147}, Pm^{148}, Pm^{148M}, Pm^{149}, Sm^{147}, Sm^{148}, Sm^{149}, Sm^{150}, Nd^{146}, Nd^{147}, Nd^{148}, B^{10}, B^{11}, Xe^{135}, I^{135}) notamment le xénon 135 et l'iode 135 ce qui permet d'effectuer une éventuelle correction sur le paramètre xénon (voir la section 3.3.1).

3.1.4 Les bibliothèques multiparamétrées

Les bibliothèques MULTICOMPO et DKLIB sont générées par le même script DRAGON4 à partir des mêmes sections. La vérification la plus en amont possible de la bifurcation (DRAGON4 $\rightarrow$ DONJON4 vs DRAGON4 $\rightarrow$ COCAGNE) consiste à vérifier les valeurs numériques stockées dans chacune des bibliothèques. Néanmoins l'information y est stockée différement.

Un programme avec interface graphique permet de visualiser le contenu d'une DKLIB, ce qui rend possible la recherche systématique d'une section efficace à paramètres donnés.

Pour visualiser une MULTICOMPO, il est seulement possible de la sortir sous format ASCII. Il n'est donc pas aisé de lire les informations d'une MULTICOMPO sous format ASCII, en effet une MULTICOMPO est organisée comme on peut le voir sur la figure 3.5.

Chaque calcul élémentaire est localisé dans la liste 'CALCULATIONS' par un tuple de paramètres globaux et/ou locaux. Les indices de ces tuples sont stockés dans un arbre dont la profondeur est égal aux nombres de paramètres globaux et locaux. Un exemple est donné figure 3.6

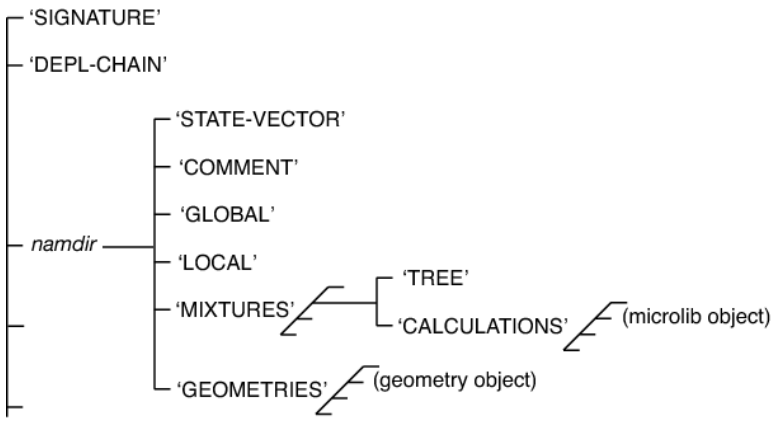


Figure 3.5 Agencement d'un objet `multicompo`

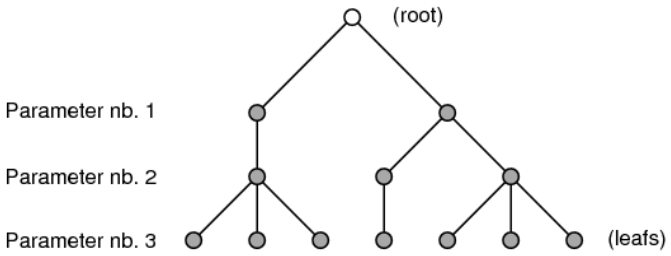


Figure 3.6 Arbre des paramètres globaux dans un objet SAPHYB

La comparaison des bibliothèques MULTICOMPO et DKLIB doit se faire pour chaque section efficace de chaque tuple de paramètres. À l'aide de l'algorithme décrit dans les pages 114-115 du rapport IGE-295 (Hébert *et al.* (2009)), un programme avec interface graphique (Toueg (2010)) a été réalisé pour visualiser le contenu d'une MULTICOMPO.

La MULTICOMPO stocke les sections efficaces corrigées par le coefficient de Selengut alors que la DKLIB stocke les sections efficaces non corrigées plus le coefficient de Selengut. À cette différence près, les valeurs sont les mêmes.

3.2 Le calcul de cœur

Maintenant que nous avons les bibliothèques de sections macroscopiques MULTICOMPO et DKLIB, nous allons comparer les grandeurs physiques en sortie des codes de cœur DONJON4 et COCAGNE.

Étant donnant la complexité de la géométrie du cœur entier (Le Mer (2007)), une concordance *a priori* entre les deux codes est improbable. Non seulement des écarts sur les résultats sont attendus pour les premiers essais, mais en plus on ne peut occulter l'éventualité qu'ils se compensent.

Afin de décorréler les éventuelles erreurs, une démarche analytique a été choisie. Nous comparerons les grandeurs physiques sur des géométries simplifiées que nous complexifierons de plus en plus jusqu'à atteindre le cœur du REP de la longueur de campagne. Une fois les comparaisons validées en statique, une évolution du combustible sera simulée (module TINST: de DONJON4). Pour finir un calcul sera effectué sur toute la longueur de campagne.

3.2.1 Combustible homogène infini

Géométrie

La première géométrie est la plus simple : du combustible homogène infini. Dans les deux codes, il sera décrit par un cube aux conditions aux limites de réflexion. Les méthodes de résolutions et le maillage sont les mêmes.

Résultats

Nous allons comparer DONJON4 et COCAGNE en terme de réactivité et de rapport de flux groupe 1 sur groupe 2. Nous comparons les rapports des flux et non pas les flux eux même car ils dépendent de la normalisation qui est propre à chaque code.

Dans les deux cas les rapports sont uniformes (milieu homogène infini) et égaux à 10^{-3} près.

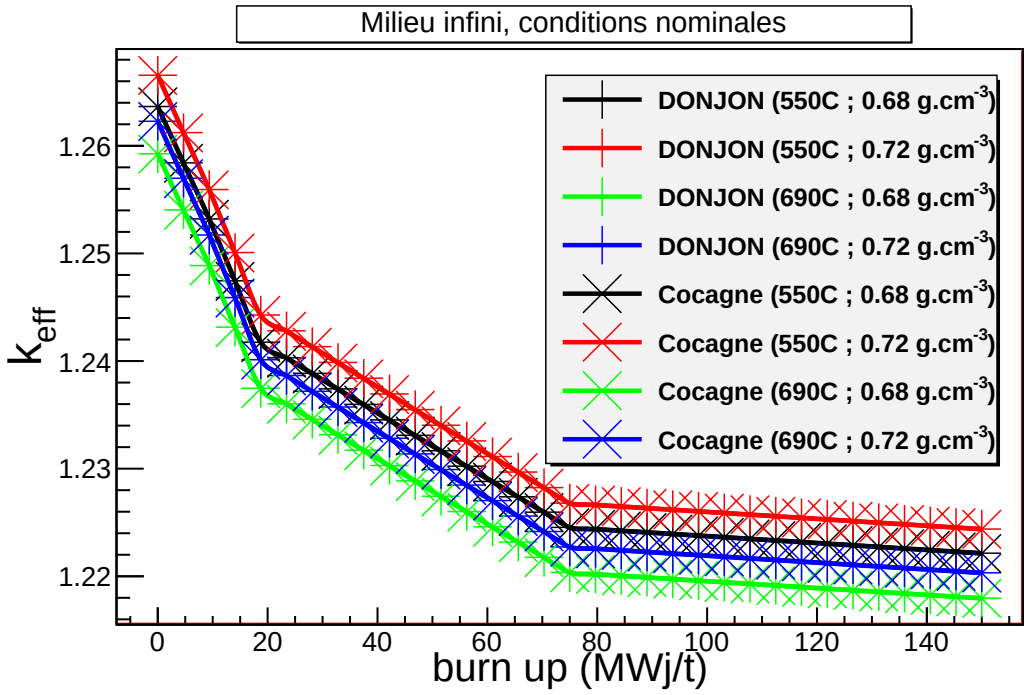


Figure 3.7 Combustible homogène infini : réactivité

En terme de réactivité, les résultats obtenus sont présentés sur la figure 3.7. La réactivité y est tracée en fonction du burnup. Chacune des quatre couleurs correspond à une certaine valeur du tuple (température combustible, densité modérateur, niveau de xénon, concentration de bore). La légende détaille les valeurs qui ne sont pas à leur niveau nominal. Seuls les coordonnées marquées par une croix ont été effectivement calculées. Les croix droites indiquent un calcul DONJON4 tandis que les croix obliques indiquent un calcul COCAGNE.

Analyse

La cohérence des deux codes se visualise bien par la superposition des marqueurs « croix droite » et « croix oblique » qui donnent l'impression d'un marqueur en forme d'étoile. Cette cohérence est d'autant plus satisfaisante que les courbes rouge, verte et bleu correspondent à des tuples qui ne sont pas stockés dans la bibliothèque macroscopique et que chacun des codes de cœur a dû interpoler les sections de la bibliothèque.

3.2.2 Combustible hétérogène

Géométrie

La deuxième géométrie est plus complexe car elle fait intervenir un deuxième milieu : du combustible homogène entouré par du réflecteur.

Dans les deux codes, elle est décrite par une même géométrie cartésienne 3D découpée régulièrement 60 fois suivant Z et composée de cellules carrées dans le plan XY. Dans ces plans les rangées extrêmes sont du réflecteur tandis que l'intérieur est composé de 30×30 cellules combustibles. Les conditions aux limites sont des conditions de flux nul.

Les méthodes de résolutions sont choisie de manière à obtenir des résultats convergé à 10^{-5} près en k_{eff} tout en minimisant l'effort de calcul, à savoir MCFD 2 (Hébert (1987a)) pour DONJON, SPN d'ordre 1 RT0 pour COCAGNE.

Résultats

En terme de réactivité, les résultats obtenus sont présentés sur la figure 3.8 suivant la même logique que la figure 3.7 (seuls les valeurs des paramètres choisis sont différents).

Pour le flux, les nappes obtenues ont la même allure que les distributions de burnup décrites à la section 3.2.3.

Analyse

Comme à la section 3.2.1, la cohérence des deux codes se visualise bien par la superposition des marqueurs. Les commentaires sur le flux seront fait en même temps que le burnup (voir la section 3.2.3).

3.2.3 Combustible hétérogène avec évolution

Géométrie

La géométrie est la même que précédemment (section 3.2.2) : du combustible homogène entouré par du réflecteur. Ce test va permettre de vérifier la pertinence du module TINST: créé pour ce mémoire.

Résultats

Partant du calcul décrit précédemment (section 3.2.2), les 30×30×60 cellules combustibles brûlent jusqu'à une moyenne de 150 MWj/t. Nous allons comparer la distribution du burnup cellule par cellule entre DONJON4 et COCAGNE.

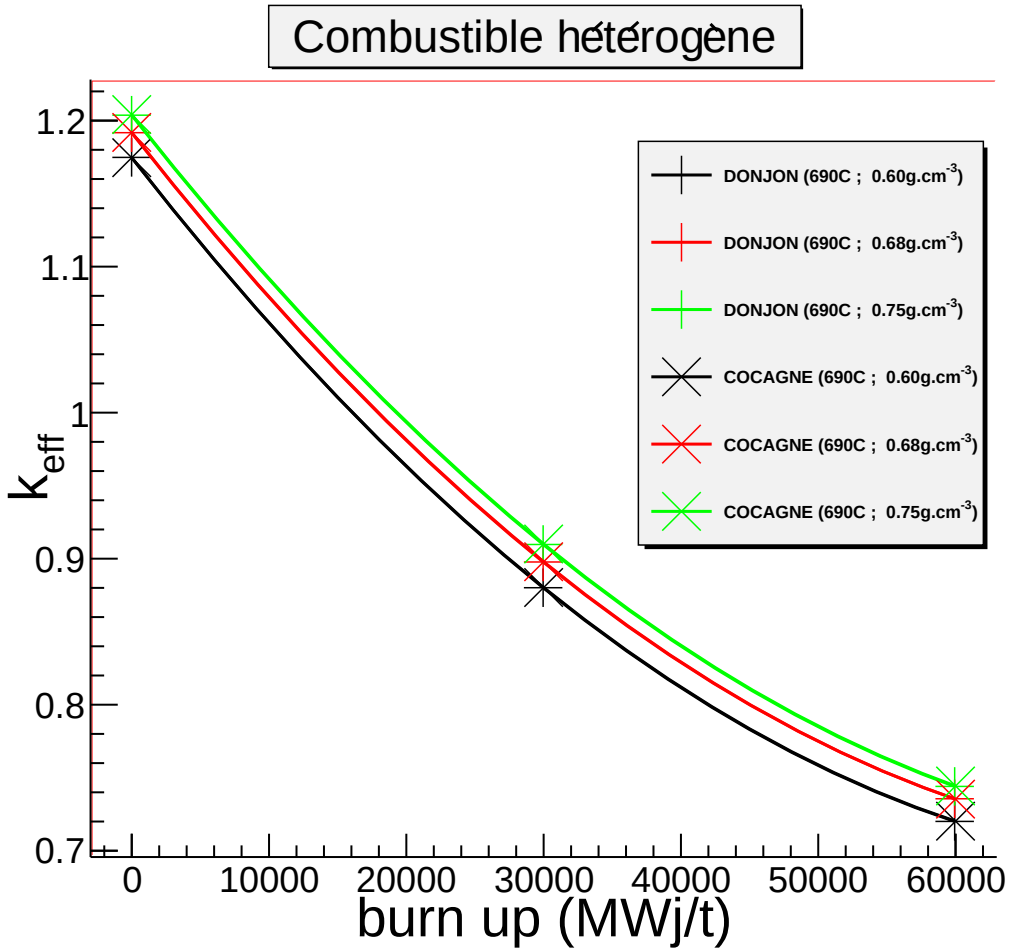


Figure 3.8 Combustible hétérogène : réactivité

Les figures 3.9 et 3.10 représentent la distribution du burnup en MWj/t (COCAGNE) dans les plans centraux XY et XZ. Les figures 3.11 et 3.12 représentent le rapport des burnups en pourcentage (DONJON4/COCAGNE) cellule par cellule dans les plans XY extrême et central. Les figures 3.13 et 3.14 représentent le rapport des burnups en pourcentage (DONJON4/COCAGNE) cellule par cellule dans les plans XZ extrême et central.

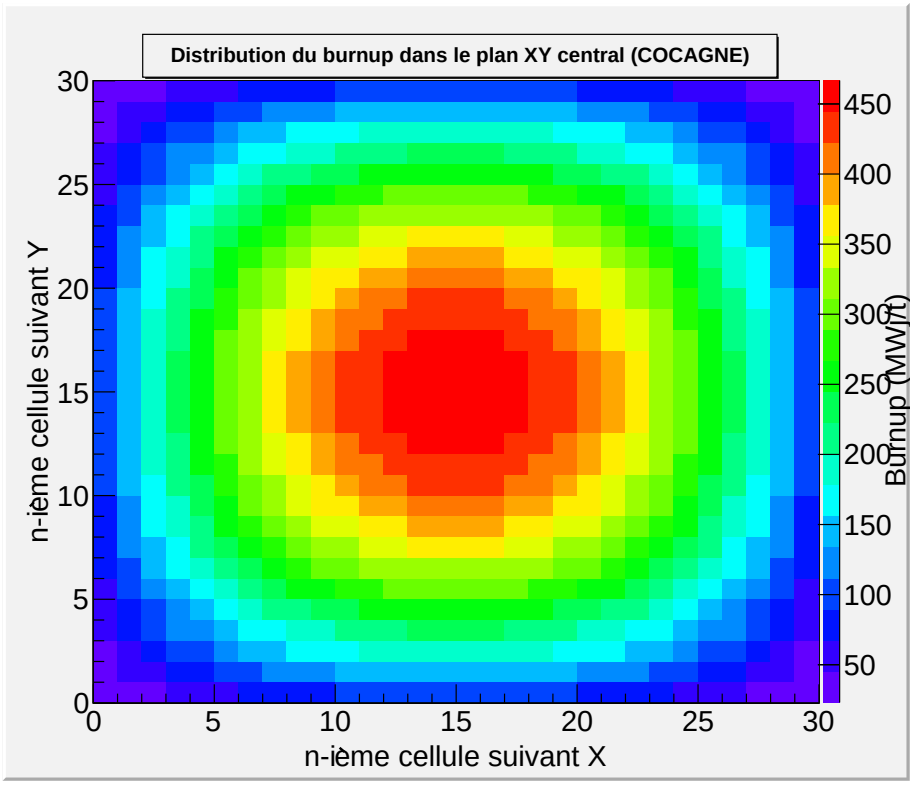


Figure 3.9 Distribution de burnup dans le plan XY central

Analyse

Les cellules représentées sur les figures à 3.9 à 3.14 ont mêmes volumes et contiennent la même mixture dans les mêmes conditions. La distribution du burnup à 150 MWj/t en moyenne a été obtenue en partant d'une situation initiale à burnup nul. Comme le module TINST: utilise le flux du calcul de diffusion pour déterminé l'irradiation, on s'attend à obtenir une distribution de burnup identique à celle du flux, ce que nous confirme l'allure typique des figures 3.9 et 3.10.

Les résultats sous forme de pourcentage (figures 3.11 à 3.14) présentent les écarts les plus importants sur les cellules extrèmes de la géométrie, *id est* celles situées au plus proche du réflecteur. Ces cellules étant excentrées, elle ne présentent pas une forte irradiation (cf.

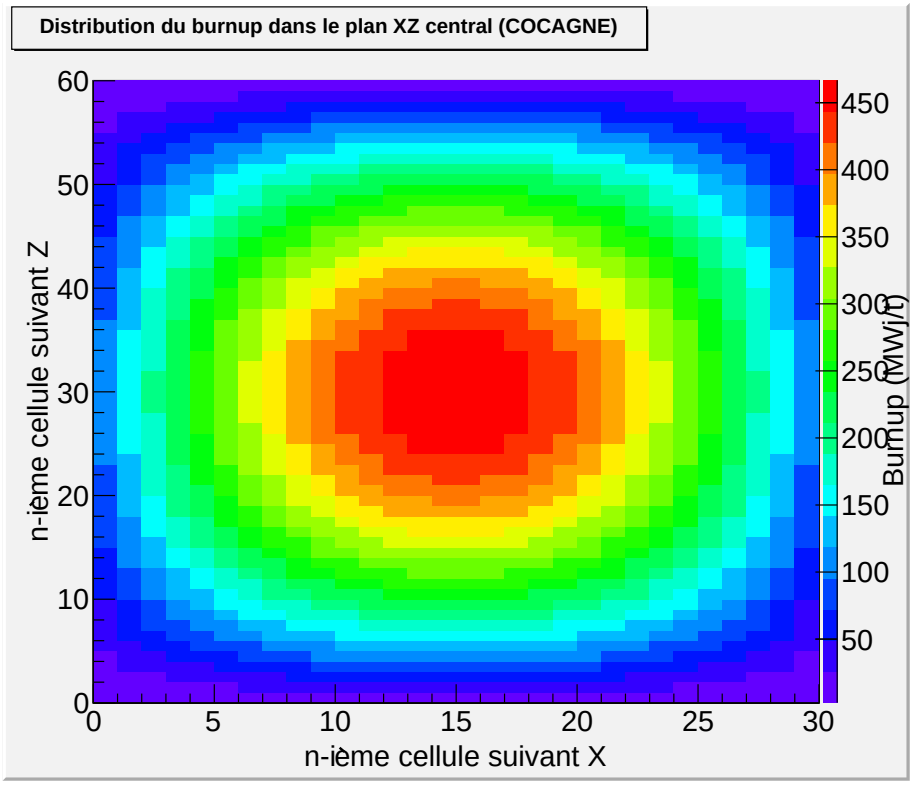


Figure 3.10 Distribution de burnup dans le plan XZ central

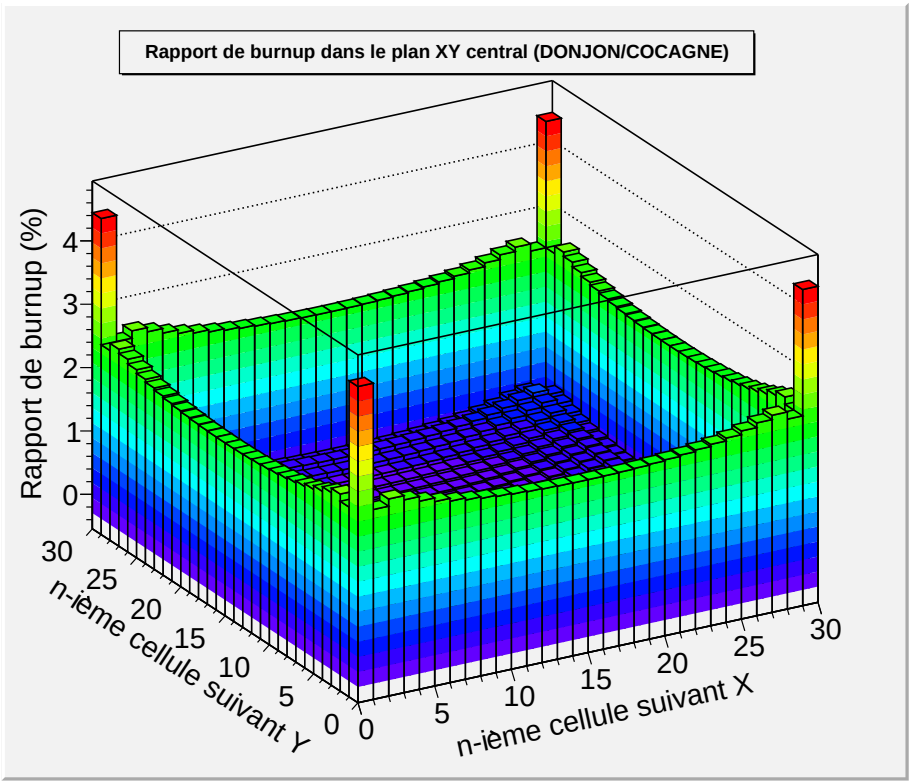


Figure 3.11 Distribution du rapport des burnups dans le plan XY central

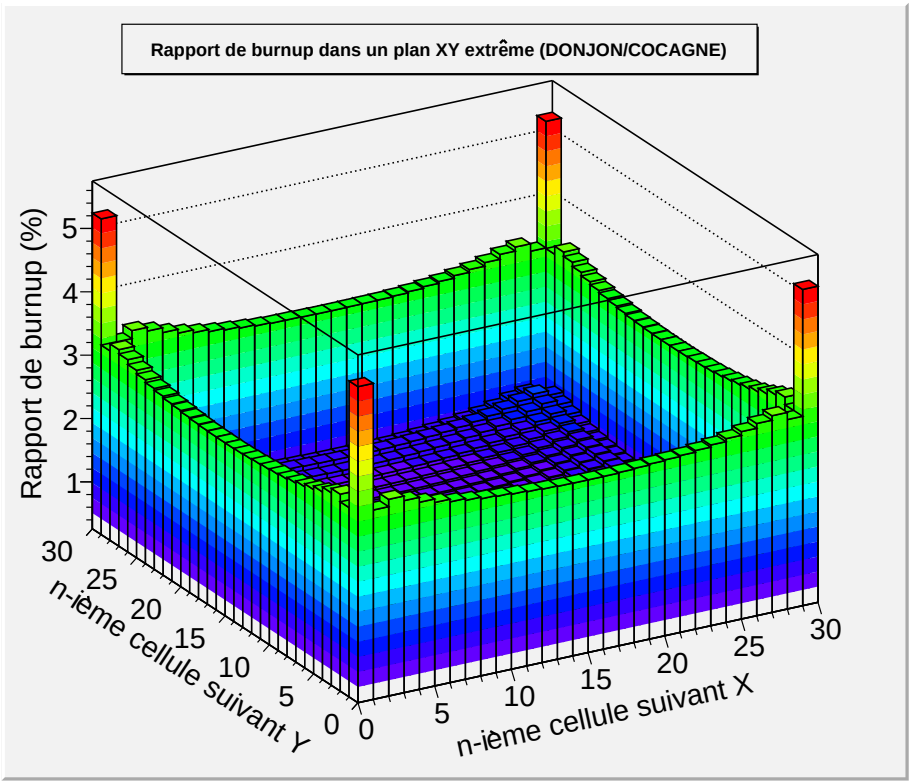


Figure 3.12 Distribution du rapport des burnups dans le plan XY extrême

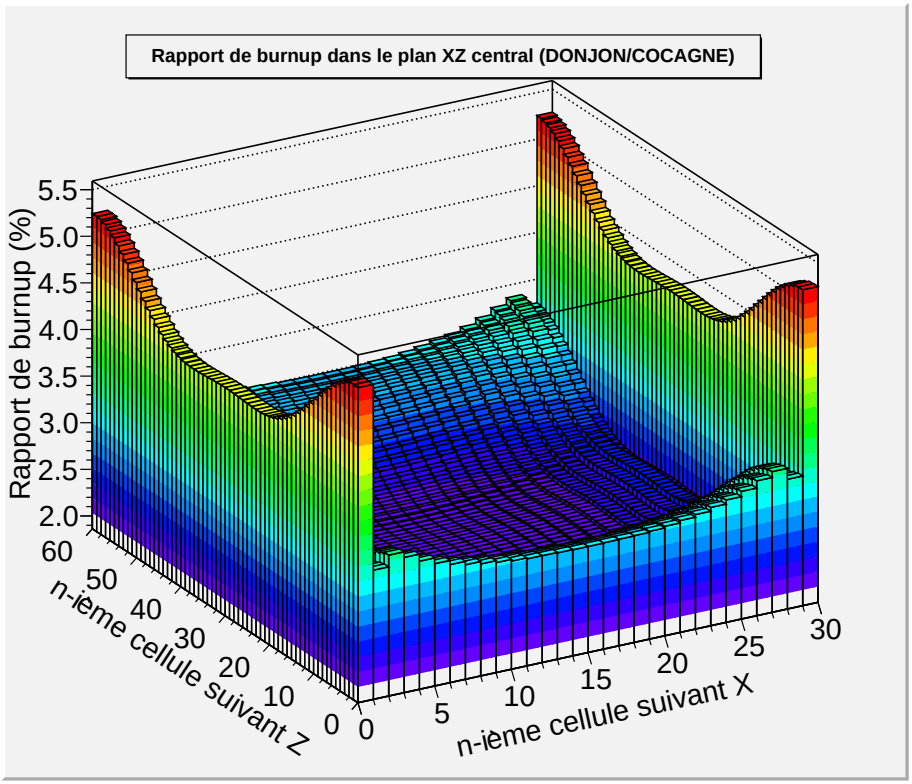


Figure 3.13 Distribution du rapport des burnups dans le plan XZ central

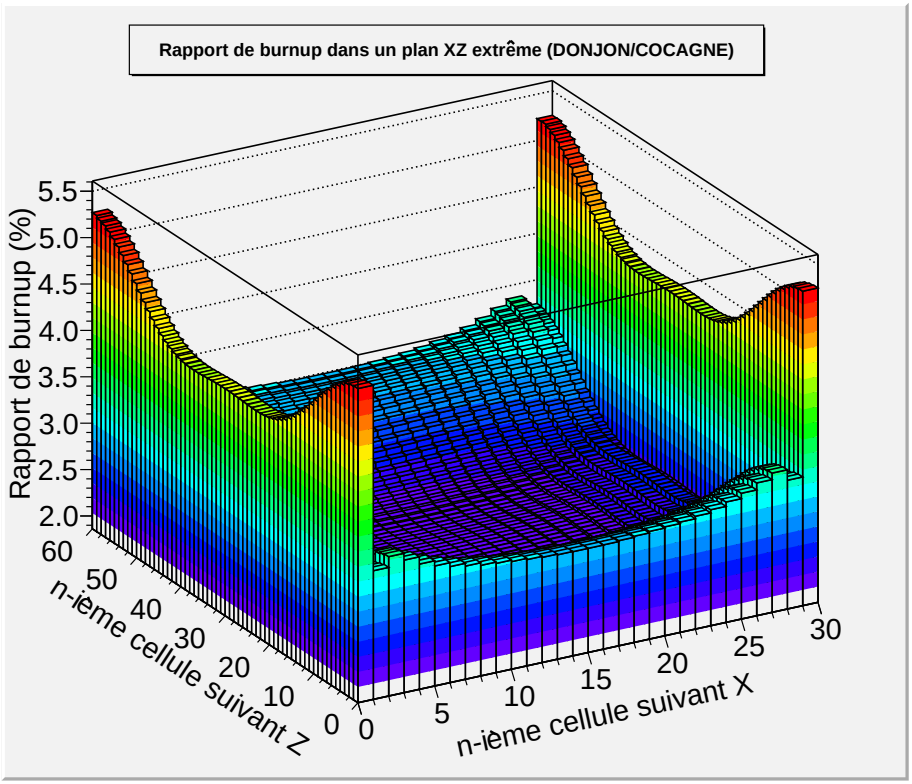


Figure 3.14 Distribution du rapport des burnups dans le plan XZ extrême

figures 3.9 et 3.10), ce qui explique que les écarts sont plus marqués qu'au centre. Pour cette même raison, le burnup moyen est identique dans DONJON4 et dans COCAGNE bien que l'intuition visuelle nous laisse croire autrement.

La forme globale de la distribution de burnup concorde entre DONJON4 et COCAGNE. De plus, les diffences numériques cellule par cellule peuvent s'expliquer par la différence de méthode numérique entre DONJON4 et COCAGNE (cf. section 3.2.2), et celles-ci sont acceptables car elles sont inférieures au pourcent dans les zones où l'irradiation est élevée (zones qui déterminent le comportement physique du réacteur). Néanmoins l'écart se fait beaucoup plus marqué à l'interface réflecteur–combustible qu'à l'interface combustible–flux nul (cf. figures 3.13 et 3.14).

3.2.4 Longueur de campagne

Géométrie

La géométrie est celle du réacteur au complet : un REP typique d'EDF. Ce test va permettre de vérifier la cohérence des codes DONJON4 et COCAGNE sur la géométrie complète du réacteur et sur toute la longueur de campagne.

Le réacteur est décrit par une géométrie cartésienne 3D. Cette géométrie est découpée en 17 × 17 cellules carrées dans le plan XY (plan horizontal) et 28 découpages axiaux. Les 20 découpages du centre contiennent du combustibles et du reflecteur tandis que les 8 découpages extérieurs ne contiennent que du réflecteur. Dans chacun des 20 plans centraux se trouvent 159 cellules combustibles (1.1). Une cellule combustible n'est rien d'autre qu'une section d'assemblage, chaque assemblage étant découpé en 20 sections de même hauteur (contrainte du module RESINI : de DONJON4). Ce découpage permet d'associer 20 tuples de la bibliothèque macroscopique pour chaque assemblage.

Distribution des paramètres

Une longueur de campagne est la durée pendant laquelle un réacteur est exploité entre deux rechargements. Pour cette longueur de campagne typique, chaque assemblage possède un burnup intial homogène (0 ; 13000 ; 26000 ou 39000MWj/t). Les assemblages sont distribués suivant une carte de burnup qui respecte une symétrie un huitième pour une moyenne d'environ 19000MWj/t au démarrage.

La simulation de la longueur de campagne consiste à faire évoluer le combustible vers un nouveau burnup moyen tout en jouant sur les valeurs des paramètres de la bibliothèque macroscopique pour assurer la criticité du cœur.

L'excès initial de réactivité est compensé par les mécanismes de contre-réaction naturels

(température, densité) et artificiels (ajout de bore dans le liquide refroidissement). Alors que COCAGNE possède un modèle thermique capable d'ajuster les températures et densités dans chaque section d'assemblage, DONJON4 en est dépourvu. À fin de comparaison, une longueur de campage a été effectué à températures et densités uniformément réparties.

Une fois la longueur de campagne validée sur un cœur à température et densité homogène, une distribution statique moyenne de ces valeurs a été choisie. En effet, après un régime transitoire de courte durée (quelques heures), la distribution de température et de densité se stabilise en régime permanent (durée d'une longueur de campagne). La distribution statique choisie est la moyenne des distributions déterminées par COCAGNE (avec modèle thermique actif) à l'issue de chaque pas de burnup. Cette distribution est ensuite introduite dans la définition du réacteur à l'instant initial et une comparaison est effectué entre DONJON4 et COCAGNE (modèle thermique désactivé).

Résultats

La figure 3.15 représente la concentration de bore en ppm (partie pour million) en fonction du burnup moyen du cœur. La courbe rouge avec marqueurs « croix oblique » correspond à DONJON4 tandis que la courbe verte avec marqueurs « croix droite » correspond à COCAGNE. L'écart obtenu est de l'ordre du ppm en concentration de bore à chaque pas de burnup.

La figure 3.16 représente le facteur de point chaud tel que défini dans la documentation de DONJON4 (Sekki *et al.* (2009)) en fonction du burnup moyen du cœur. Le facteur de point chaud est le maximum du rapport de la puissance d'une section d'assemblage sur la puissance moyenne du cœur. La courbe rouge avec marqueurs « croix oblique » correspond à DONJON4 tandis que la courbe verte avec marqueurs « croix droite » correspond à COCAGNE. L'écart obtenu est de l'orde de 10^{-4} à chaque pas de burnup.

Analyse

On peut apprécier sur les deux figures précédentes la cohérence DONJON4 et COCAGNE sur une longueur de campagne au complet. Cette cohérence est d'autant plus satisfaisante qu'elle reste constante d'un pas de burnup à l'autre : il n'y a pas de phénomène d'accumulation d'écart.

Le facteur de point chute brutalement au début de la campagne. Bien que nous obtenions une cohérence entre les codes, cette chute brutale n'est pas conforme à l'expérience. Cependant le modèle thermique est ici désactivé, et la distribution des paramètres au sein du cœur (température, densité) a été choisie en moyenne sur toute la longueur de campagne, ce qui

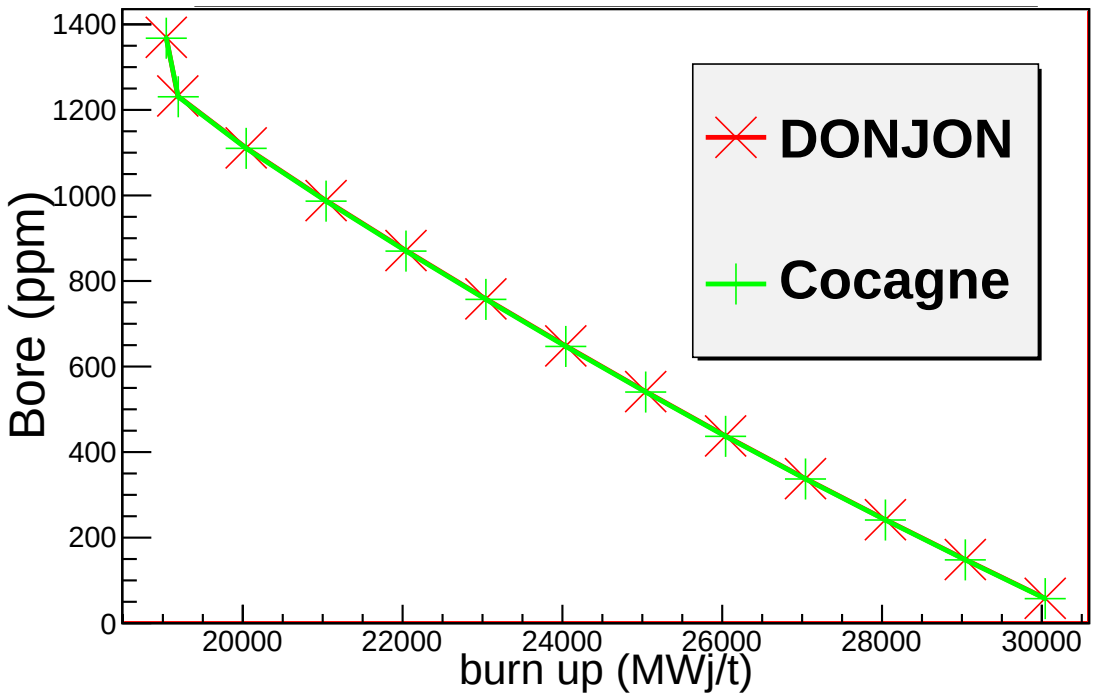


Figure 3.15 Longueur de campagne : concentration de bore critique

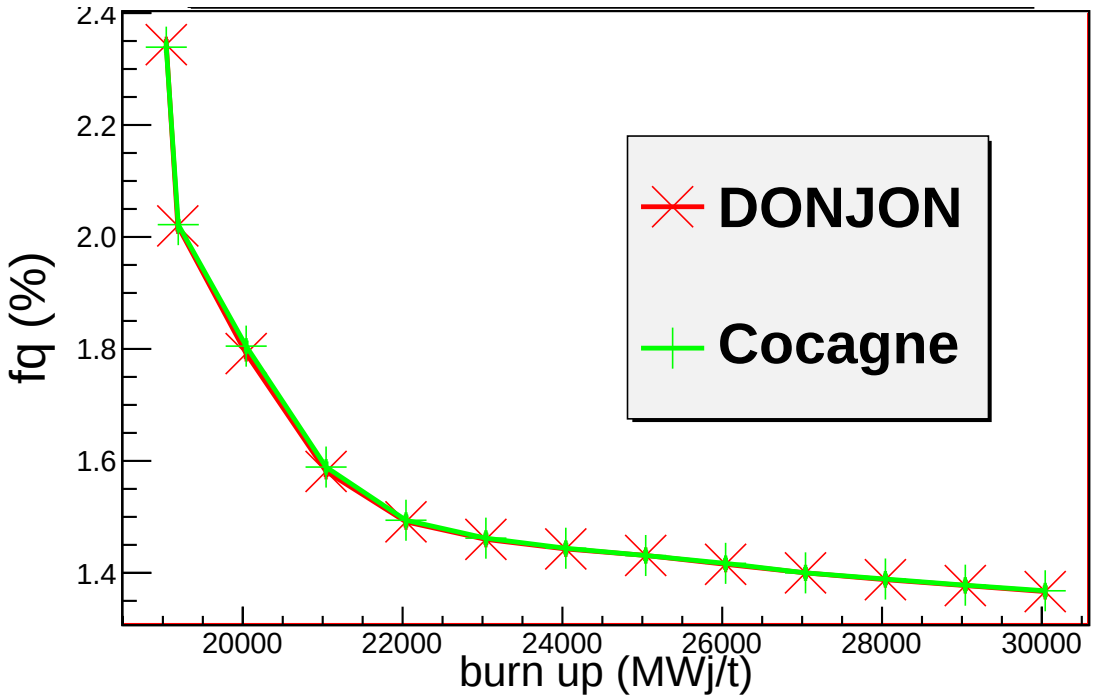


Figure 3.16 Longueur de campagne : facteur de point chaud

implique que l'écart avec le cas du modèle thermique activé est le plus fort au démarrage.

3.3 Conclusion

Nous avons vu que les codes DONJON4 et COCAGNE sont cohérents. Toutefois des difficultés ont été surmontées pour obtenir cette cohérence. Certaines d'entre elles étaient des bogues (qui ont été résolus), d'autres des différences de comportement entre les code de cœur. Ces dernières méritent que l'on s'y attarde, notamment la prise en compte de l'effet xénon et la méthode d'interpolation des bibliothèques.

3.3.1 Effet xénon

Dans un réacteur nucléaire thermique, la fission produit de l'Iode 135, qui se dégrade avec un retard de quelques heures en xénon 135. Le xénon 135 a la plus grande section efficace connue dans le domaine des neutrons thermiques (3 millions de barns), ce qui en fait un poison neutronique ayant un impact sur la réactivité du système. Du fait de sa section efficace importante, le xénon 135 absorbe des neutrons par capture, d'où le terme « empoisonnement xénon » ou « effet xénon ». En fonctionnement stable, cet empoisonnement est proportionnel au flux neutronique du cœur. Il est de l'ordre de 3000 pcm pour un réacteur REP d'EDF, et n'apparaît que quelque temps après la divergence.

Dans la bibliothèque macroscopique, le paramètre xénon ne possède que deux valeurs : 0 (absence de xénon) ou 1 (xénon saturé). Quand le paramètre vaut 1, cela signifie que la concentration de xénon dans le combustible correspond à la concentration en fonctionnement stable. Dans le calcul de fil avec DRAGON4 (paragraphe 3.1.2), le combustible vierge de xénon commence à brûler dès l'instant 0^+ au niveau de flux nominal, ce qui induit un effet xénon prononcé. Or dans la pratique un réacteur n'est jamais démarré à froid : les conditions d'exploitation sont telles qu'on commence par atteindre la divergence pour des niveaux de flux peu élevé pour ensuite monter en régime, par conséquent l'effet xénon ne se fait pas ressentir de façon brutale au démarrage.

La figure 3.17 représente la réactivité du combustible homogène infini présenté au paragraphe 3.1.2 en fonction du burnup. La courbe rouge correspond au xénon saturé dans la bibliothèque alors que la courbe verte correspond au cas du xénon nul. On peut voir qu'en absence de xénon, la réactivité du combustible diminue linéairement et en pente douce, ce que l'on peut attribuer à une perte de réactivité due uniquement à l'épuisement de noyaux fissiles. Avec le xénon à saturation, la réactivité baisse subitement jusqu'à 70MWj/t,

Pour les faibles valeurs de burnup, le code COCAGNE effectue une correction sur le paramètre xénon. Cette correction, dite « xénon équilibre », consiste à estimer une nouvelle

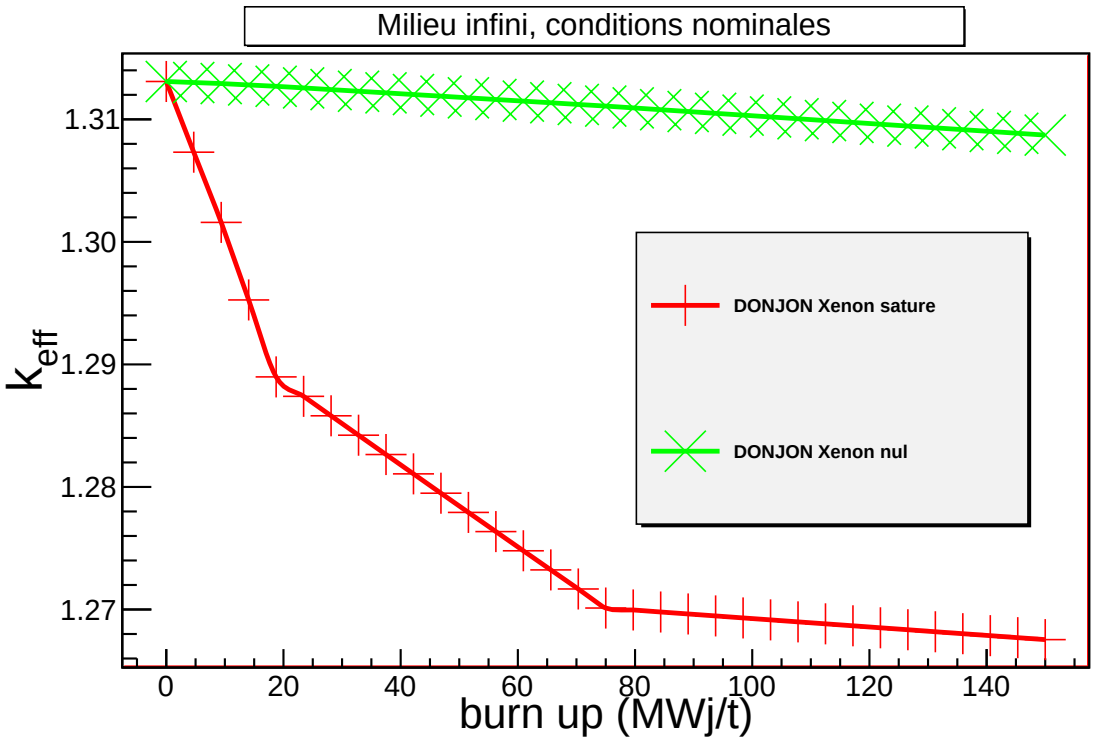


Figure 3.17 Xénon nul

valeur du paramètre de xénon (entre 0 et 1) afin de ne pas ressentir la forte perte de réactivité xénon au démarrage du réacteur. Ainsi quand le calcul de cœur ne débute pas à xénon nul, ni à xénon saturé (nul à 0MWj/t à cause du calcul de fil) mais à une valeur intermédiaire, même à 0MWj/t. Le mise en œuvre de cette correction nécessite la particularisation du xénon lors de la création de la bibliothèque macroscopique.

La figure 3.18 représente la réactivité du combustible homogène infini présenté au paragraphe 3.1.2 en fonction du burnup. La courbe rouge correspond au xénon saturé dans la bibliothèque alors que la courbe bleue représente la valeur corrigée de xénon. DONJON4 ne sait pas faire cette correction.

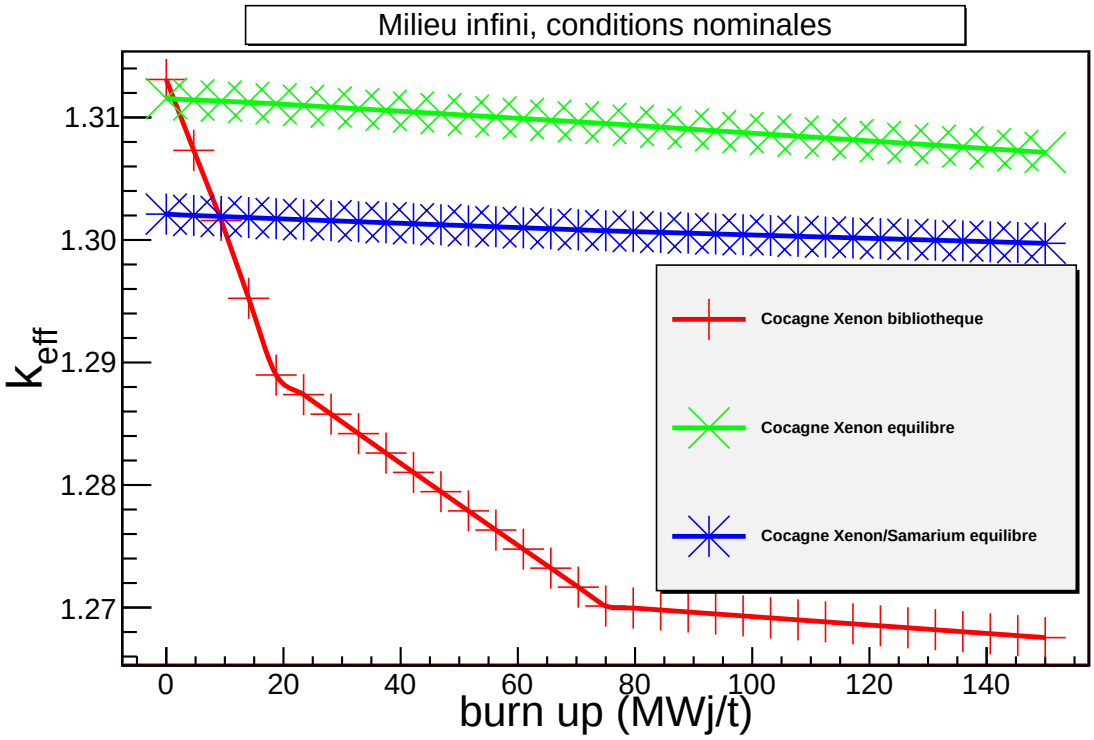


Figure 3.18 Xénon équlibre et bibliothèque

3.3.2 Interpolation des sections efficaces

Description

Lorsque que l'on a besoin d'une section macroscopique pour un tuple qui n'est pas explicitement dans la bibliothèque, le code de cœur effectue une interpolation à partir de plusieurs tuples voisins. Cette étape est importante au moins pour deux raison : d'une part

la qualité de l'interpolation influe sur la précision du calcul, d'autre part l'interpolation représente l'étape la plus gourmande en ressource ; d'où la nécessité d'un bon compromis.

Alors que COCAGNE ne sait qu'effectuer une interpolation linéaire à partir des tuples les plus proches, DONJON4 est capable d'effectuer une interpolation cubique des sections de la bibliothèque macroscopique tout en s'assurant que la courbe interpolante est à dérivée continue (Hébert (2010)).

Afin de déterminer quelle interpolation est la plus appropriée, nous allons calculer la réactivité du combustible homogène infini décrit à la section 3.2.1.

Résultats

La figure 3.19 représente la réactivité en fonction du burnup pour une plage de burnup de faible valeur (0 à 150MWj/t). Les courbes rouge et verte sont toutes deux des calculs DONJON4 issus de la même bibliothèque macroscopique, interpolée linéairement pour l'une et de manière cubique pour l'autre. Les valeurs des paramètres autres que le burnup sont fixées à leur valeur nominale. La figure possède en abscisse 32 points de burnup équirépartis de 0 à 150MWj/t dont seuls 5 points sont explicitement dans la bibliothèque. Un troisième calcul DONJON4, en bleu, a été mené avec une bibliothèque macroscopique exhaustive. Cette bibliothèque a été réalisée avec le même calcul de fil que celui décrit à la section 3.1.2. Pour cette courbe tous les points de burnup sont dans la bibliothèque et il n'y a donc aucune interpolation.

Analyse

La première chose à remarquer sur la figure 3.19 est la superposition parfaite des réactivités pour les 5 points présents dans la bibliothèque. Ce résultat attendu confirme la cohérence des deux calculs de fil ayant générer les bibliothèques macroscopiques de ce test.

À première vue, on peut apprécié le fait que l'interpolation cubique (courbe verte) produit en moyenne un meilleur résultat quantitatif, notamment sur la plage de burnup 18,75MWj/t-75MWj/t où l'interpolation linéaire (courbe rouge) surestime de plus de 500pcm le calcul de référence (courbe bleue). Néanmoins l'interpolation cubique possède un mauvais comportement qualitatif : à partir d'environ 60MWj/t la réactivité croit de presque 100pcm jusqu'à un burnup de 95MWj/t, ce qui est préjudiciable même s'il reste difficile d'en évaluer l'a répercussion sur un calcul de cœur complet. Alors que la concordance entre l'interpolation linéaire et la référence est tout à fait appréciable au delà de 75MWj/t, l'interpolation cubique s'en sort assez mal. Le tracé de la courbe s'arrête à 150MWj/t car au delà, les deux interpolations sont équivalentes.

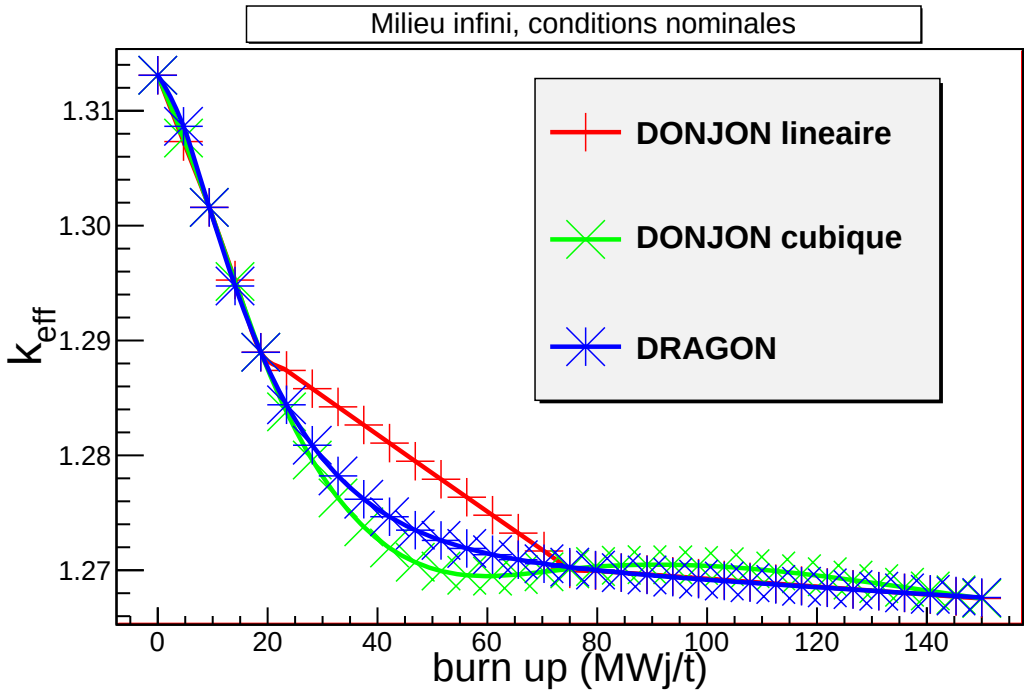


Figure 3.19 Interpolation linéaire, cubique ou pas d'interpolation

La pertinence de l'interpolation est liée aux choix des points de la bibliothèque. Néanmoins, le code COCAGNE peut se passer de l'interpolation cubique grâce à sa prise en charge de l'effet xénon (section 3.3.1). En effet, l'erreur entre l'interpolation linéraire et pas d'interpolation est due à la chute brutal de réactivité dans les faibles burnup. Le fait d'imposer le xénon à l'équilibre dès le départ permet de contourner ce problème sur la plage de burnup 18,75MWj/t-75MWj/t.

3.3.3 Symétrie numérique

Dans un calcul à géométrie cartésienne 3D telle que décrit à la section 3.2.2, il est remarquable que COCAGNE obtienne des résultats numériques identiques dans des cellules symétriques. Sur la figure 3.20 (rapport des plans symétriques calculés par DONJON4 dans le cas correspondant à la figure 3.9) apparaissent des écarts de l'ordre des erreurs de calcul numérique (pour dix-mille), tandis que COCAGNE ne présente absolument aucun écart.

Bien que n'ayant aucun impact sur le comportement du modèle, cela indique une bonne optimisation du solveur de flux de COCAGNE dans les cas symétriques.

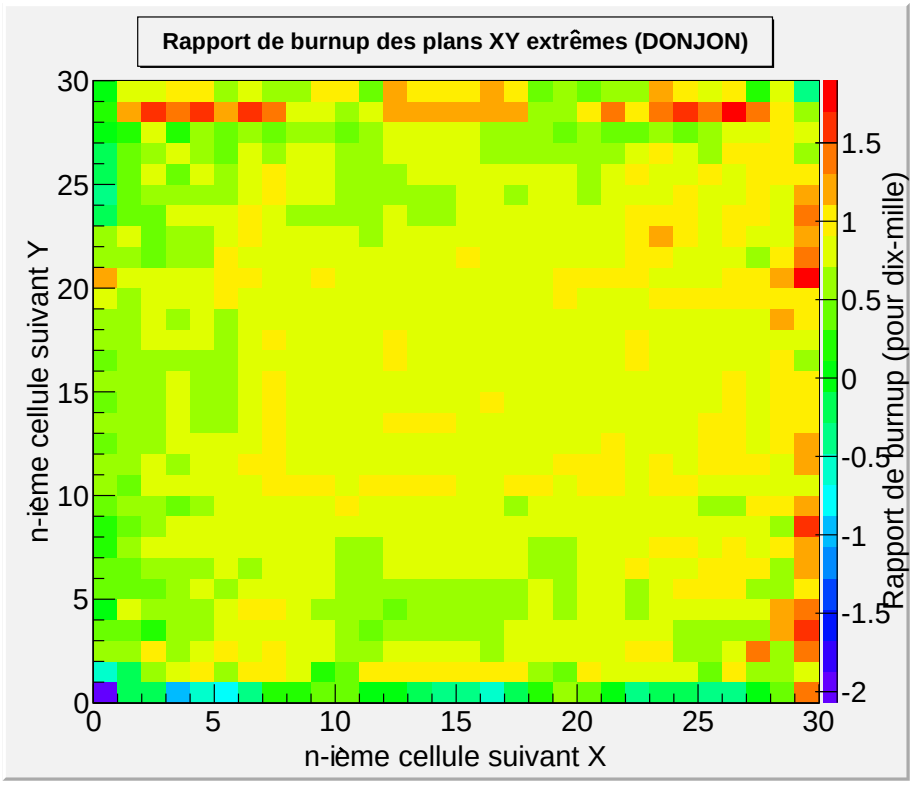


Figure 3.20 Rapport de burnup entre les deux plans symétriques extrêmes

CHAPITRE 4

CAMPAGNE AVEC EFFETS D'HISTORIQUE

L'objectif de cette partie est d'effectuer à l'aide de DONJON4 deux calculs de longueur de campagne d'un REP, l'un avec effet d'historique, l'autre sans, afin de déterminer les avantages et les inconvénients des deux méthodes.

Dans un premier temps, nous parlerons du contexte technique en terme d'adaptation du code, d'options de compilation et d'utilisation de la version MPI des codes DRAGON4 et DONJON4.

Ensuite nous lancerons les calculs d'historique et de bibliothèque, d'abord dans des conditions nominales uniformes de température et de densité puis suivant une distribution statique typique d'un REP au cours d'une longueur de campagne. Le résultats seront comparés et analysés.

Nous conclurons sur une discussion à propos de la pertinence de l'effet d'historique pour modéliser un cœur de REP.

4.1 L'environnement technique

4.1.1 Le cluster Clamart2

Pour ce travail de maîtrise, EDF a mis à disposition un accès à son cluster de calcul Clamart2. S'y connecter depuis l'extérieur requiert un environnement Windows sur lequel est installé un client VPN spécifique.

L'architecture du cluster est basé sur des processeurs Intel X5570 2.93GHz. Des environnements parallèles UNIX sont disponibles à 1, 16, 32 et 64 octoprocesseurs 64 bits. La soumission d'un calcul se fait grâce au logiciel PBS (Portable Batch System) qui effectue la planification des tâches, c'est à dire qu'il répartit les travaux par lots parmi les ressources disponibles.

La politique du service informatique d'EDF consiste à inciter les utilisateurs à travailler en 64 bits. Or DRAGON4 et DONJON4 ont été conçus spécifiquement pour le 32 bits et une adaptation du code s'impose.

4.1.2 De 32 bits à 64 bits

DRAGON4 et DONJON4 sont écrits en FORTRAN77. Le FORTRAN77 ne supporte pas l'utilisation de pointeurs, ce sont donc des variables de type INTEGER qui stockent les adresses mémoires. Or en 64 bits, les entiers restent codés sur 32 bits tandis que les adresses mémoires passent de 32 à 64 bits. Le couplage entiers/pointeurs crée donc une incompatiblité en 64 bits.

Deux solutions existent : réécrire DRAGON4 et DONJON4 pour distinguer les entiers pointeurs des entiers INTEGER ou imposer la taille des entiers à 64 bits lors de la compilation. La décision qui a été prise est la deuxième, c'est-à-dire forcer le passage des entiers en entiers longs (64 bits). Une telle modification peut s'effectuer avec le compilateur *gfortran* de GNU Compiler Collection (GCC) à l'aide des options suivantes :

```
-m64 -fdefault-integer-8 -fdefault-real-8 -fdefault-double-8
```

Néanmoins, le fait de forcer les entiers en entiers longs bouleverse aussi les réels et les doubles. Par conséquent tous les conteneurs (tableaux et structures LCM) doivent être adaptés. D'autres corrections doivent aussi être apportées, comme pour permettre la lecture du fichier d'entrée CLE-2000 en format 64 bits. Une réécriture partielle du code (bien plus raisonnable que le découplage entiers/pointeurs) a donc été nécessaire. La version utilisée pour le reste de notre travail est la version 4.0.4 du 24 décembre 2010.

4.1.3 La librairie MPI

Présentation

MPI (The Message Passing Interface) (Gropp *et al.* (1996)) est une norme définissant une bibliothèque de fonctions utilisable avec les langages C et Fortran. Elle permet d'exploiter des ordinateurs distants ou multiprocesseur par passage de messages.

Elle est devenue *de facto* un standard de communication pour des nœuds exécutant des programmes parallèles sur des systèmes à mémoire distribuée. MPI a été écrite pour obtenir de bonnes performances aussi bien sur des machines massivement parallèles à mémoire partagée que sur des clusters d'ordinateurs hétérogènes à mémoire distribuée. Elle est disponible sur de très nombreux matériels et systèmes d'exploitation. Ainsi, MPI possède l'avantage par rapport au plus vieilles bibliothèques de passage de messages d'être grandement portable (car MPI a été implantée sur presque toutes les architectures de mémoires) et rapide (car chaque implantation a été optimisée pour le matériel sur lequel il s'exécute).

C'est la bibliothèque Open MPI (ope (2004)) qui a été utilisée pour permettre à DRAGON4 et DONJON4 d'exploiter les capacités du cluster d'EDF. Les modules DRVMPI: et SNDMPI: ont été écrits pour ce travail et sont disponibles dans la version 4.0.4.

Fonctionnement

MPI consiste à lancer le même exécutable simultanément sur n processeurs, n étant choisi par l'utilisateur au lancement. Cet exécutable unique possède deux variables qui ne sont connues qu'à l'exécution : le nombre n de processeurs et son rang r compris entre 0 et $n-1$. En effectuant des tests sur ces variables, il devient possible de ne pas exécuter les mêmes instructions d'un processeur à l'autre, bien que le fichier binaire du programme soit le même. Les fonctionnalités d'Open MPI permettent d'envoyer un message (par exemple sous la forme d'un buffer d'entiers) d'un processeur à l'autre.

Les modules créés pour ce travail de maîtrise (cf. Annexe B) rendent accessible ces fonctionnalités depuis le CLE-2000. En pratique, le script rdonjon (cf. Annexe C) est exécuté n fois et le rang r est récupérable dans une variable CLE-2000 via le module DRVMPI: . Des tests de type IF sur cette variable dans le script CLE-2000 permettent de choisir les blocs d'instructions que l'on souhaite rendre accessible à un processeur donné. L'envoi d'une structure LCM (LINKED_LIST ou XSM) d'un processus r à un processus s se fait par passage de message bloquant. Cela signifie r demande à s l'autorisation de lui envoyer le message, et si s n'est pas encore prêt l'exécution du programme sur r est bloqué. Il en va de même si s attend un message de r et que r n'est pas prêt à le lui envoyer. Cette contrainte est liée à l'utilisation sous-jacente de la fonction MPI_SEND. D'autres types d'envoi sont possibles

mais l'envoi de messages bloquants est simple et robuste. De toute façon la communication entre les processeurs ne constitue pas un goulot d'étranglement dans le calcul d'historique, donc ce choix est raisonnable.

La compilateur MPI du cluster Clamart2 n'est pas configuré pour fonctionner avec *gfortran*, qui est le seul compilateur permettant la compilation en 64 bits de DRAGON4 et DONJON4 (cf. section 4.1.2). Le compilateur de ce projet a été construit à partir des source d'Open MPI (`http://www.open-mpi.org/software/ompi/v1.4/`) avec les options suivantes :

```
#!/bin/bash

./configure --prefix=$HOME/openmpi --disable-mpi-f90 CC=gcc CFLAGS="-m64" F77=gfortran FC=↩
    gfortran FFLAGS="-m64" FCFLAGS="-m64"
```

Ce choix d'option a été discuté avec les utilisateurs d'Open MPI (`http://www.open-mpi.org/community/lists/users/2010/12/15029.php`). Le script de lancement rdonjon spécifique à MPI est disponible en Annexe C.

4.1.4 Templates et scripts CLE-2000

Limites du CLE-2000

La réalisation du calcul d'historique nécessite une grande quantité de LINKED_LISTs (plusieurs pour chacune des 260 mixtures de notre modèle). On aimerait pouvoir les manipuler dans le script CLE-2000 à l'aide d'un tableau de LINKED_LISTs indexé par mixture.

Mais de tels tableaux n'existent pas en CLE-2000 et il est interdit d'utiliser une variable de type STRING pour suffixer un nom de LINKED_LIST. Le langage CLE-2000 sait stocker des LINKED_LISTs dans une plus grosse LINKED_LIST, mais lire ou modifier l'une d'entre elle via l'appel à un module n'est pas possible. Il est nécessaire de créer une LINKED_LIST extraite grâce à la commande STEP UP, de l'utiliser puis d'écraser l'ancienne avec la nouvelle. Ces manipulations allourdissent considérablement l'exécution du calcul et l'occupation en mémoire.

Une solution déjà utilisée dans d'autres travaux de maîtrise (comme Le Mer (2007)) consiste à créer un script CLE-2000 avec un autre langage de programmation (perl par exemple). Nous proposons de généraliser cette approche avec un langage de template.

Un langage de template

Un template est un fichier texte générique qui produit un autre fichier texte (x2m, c2m, HTML, XML, etc.). Le template contient des variables et des étiquettes qui disparaissent

dans le fichier produit. Dans ce travail de maîtrise, les templates structurent les fichiers CLE-2000 dont le contenu est voué à changer régulièrement (comme le maillage dans le module GEO: ou les paramètres physique dans le module RESINI:). Le template permet donc de séparer la forme (agencement du maillage entre les mots clefs d'un module) du contenu (suite de réels décrivant le maillage).

Le template est aussi capable de répéter un motif (comme un paragraphe entier) et de n'en changer que quelques mots à chaque tour de boucle. Cela permet de simuler une boucle *FOR* CLE-2000 à l'intérieur de laquelle seul les noms des LINKED_LIST changent (MICLIB_01, MICLIB_02, MICLIB_03, etc.).

Le langage de template choisi est celui du framework de développement web Django réalisé en Python. Il présente l'avantage d'être utilisable avec Python, un langage simple, open source, multiplateforme et disponible sur la grande majorité des environnements Unix.

Voici quelques lignes d'un template CLE-2000 typique :

```
{% if mpi_enabled %}
MODULE SNDMPI: ;
{% for assembly in assembly_list|slice:':2' %}
XSM_FILE {{ assembly.XSMCPO }} ;
{{ assembly.XSMCPO }} := SNDMPI: {{ assembly.COMPO }} ::
               EDIT 0 FROM {{ assembly.cpurank }} TO <<cpumain>> ;
{% endfor %}
{% endif %}
```

Dans cet exemple *assembly_list* est une liste d'objets contenant les attributs *XSMCPO*, *COMPO* et *cpurank*. La syntaxe d'évaluation des variable consiste en une encapsulation par des doubles crochets $\{\{\cdots\}\}$. Les étiquettes (encapsulées à l'intérieur des symboles $\{\%\cdots\%\}$) portent des noms explicitant leur fonctions (for, if, etc.). La documentation complète se trouve à l'adresse `http://docs.djangoproject.com/en/1.2/topics/templates/`.

4.2 Le schéma de calcul d'historique

Le calcul d'historique consiste à réaliser un couplage entre le code de cœur et le code d'assemblage.

Le diagramme 4.1 montre que le calcul classique fait intervenir le code d'assemblage en amont et le code de cœur en aval. À l'inverse, on peut voir sur le diagramme 4.2 que le calcul d'historique se sert des résultats du code de cœur pour rétroagir sur le code d'assemblage.

Nous détaillerons le déroulement du schéma calcul d'historique dans l'ordre de la lecture du diagramme 4.2.

4.2.1 Les calculs de fil

Grâce à une symétrie un huitième des paramètres (températures, densités et burnups), le nombre d'assemblages modélisés se réduit de 157 à 26. À première vue, chaque assemblage est modélisé dans le module RESINI: par un bundle. Or, à chaque bundle n'est associé qu'une valeur de paramètre. Pour affiner la distribtion de ces paramètres, nous utiliserons plusieurs bundles par assemblage. Ainsi la distribution axiale des paramètres est prise en compte par les bundles qui constituent des sections d'assemblages (et non pas des assemblages entiers). Comme le montre la première ligne du diagramme 4.2, il y a autant de calculs de fil que de sections d'assemblage.

Les propriétés de chaque bundle sont obtenues par un calcul de fil qui reprends la géométrie d'un assemblage entier, comme décrit au paragraphe 3.1.1. De fait chacune des sections est modélisée comme un assemblage entier en milieu infini. Ne pas prendre en compte les effets d'environnement permet de paralléliser tous ces calculs de fil indépendants.

Chaque bundle est initialisé avec les valeurs de température et de densité qui lui sont propres. Par contre la concentration de bore reste fixée à sa valeur nominale, à savoir 500 parties pour million.

Le schéma du calcul de fil est le même que le schéma à deux niveaux décrit au paragraphe 3.1.2 :

- L'équation du transport, sur la géométrie N1 contenant 40 mixtures définies pour 281 groupes d'énergie, est résolue par un calcul de type B avec des conditions aux limites de réflexion (milieu infini). L'inventaire isotopique de ces 40 mixtures est initialisé à burnup nul.
- Le flux N1 issu du calcul précédent est utilisé pour condenser les mixtures à 26 groupes d'énergie, que l'on duplique à hauteur de 164 mixtures (stretching pour fitter la géométrie N2).
- L'équation du transport, sur la géométrie N2 contenant 164 mixtures définies pour 26

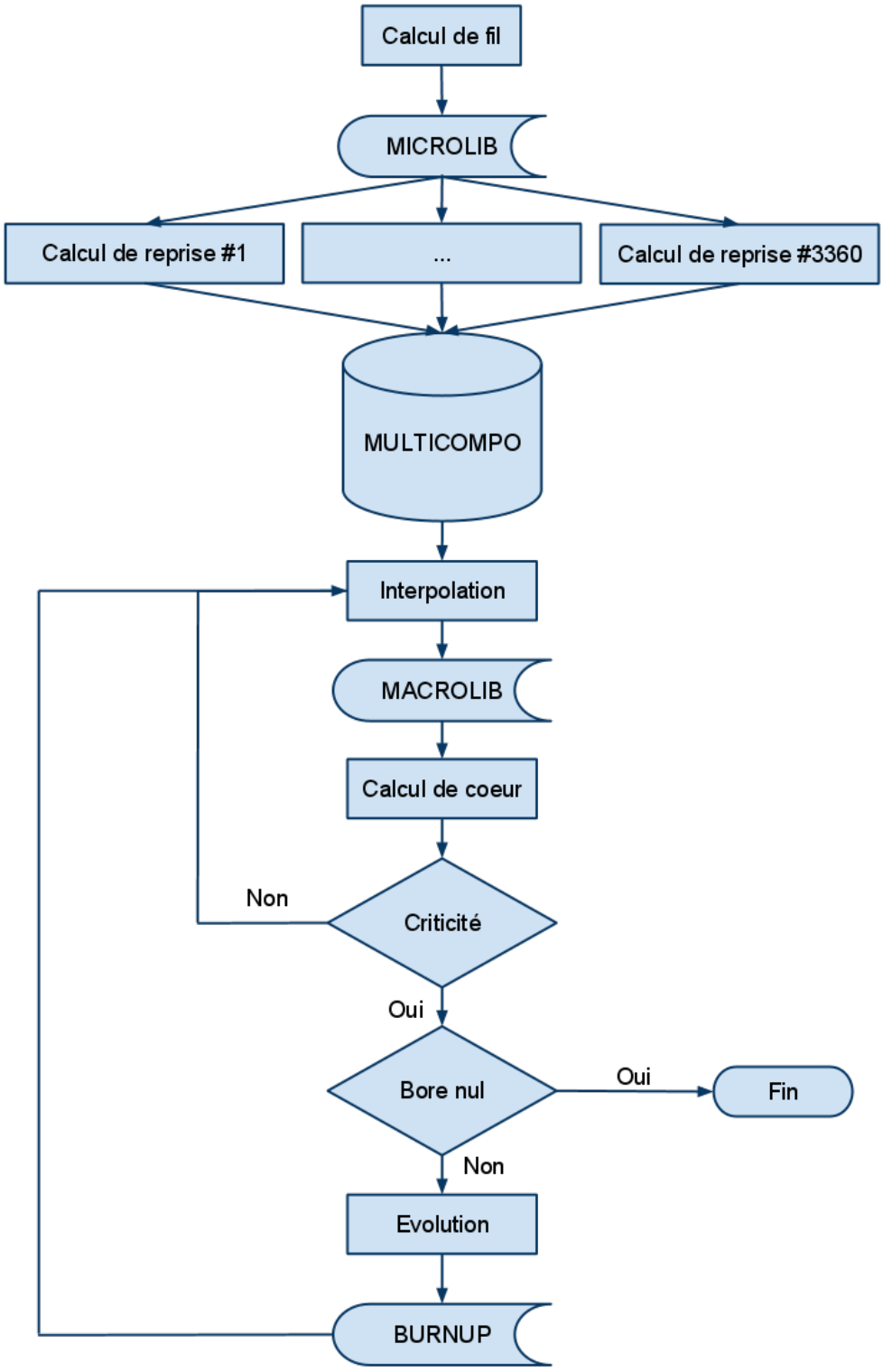


Figure 4.1 Diagramme calcul classique

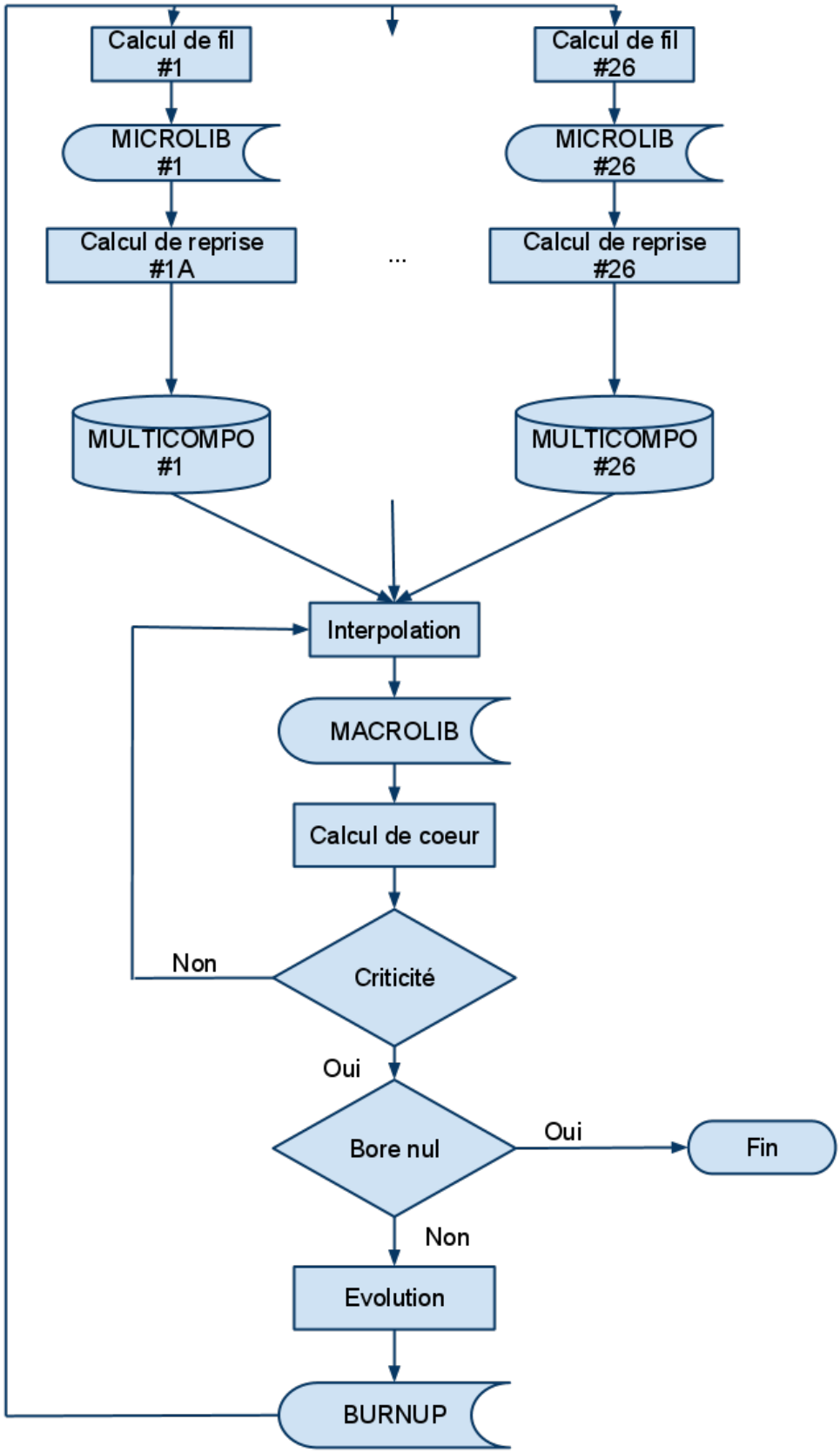


Figure 4.2 Diagramme calcul d'historique

groupes d'énergie, est résolue par un calcul de type K avec des conditions aux limites de réflexions (milieu infini).
- Le flux N1 issu du calcul précédent est utilisé pour faire évoluer les mixtures au burnup suivant.
- Les 164 mixtures sont alors condensées à 1 groupe et refusionnées pour reformer les 40 mixtures initiales, et ainsi obtenir les concentrations isotopiques recherchées.

Contrairement au calcul de fil classique le processus n'est pas répété 73 fois pour obtenir 28 concentrations, mais juste assez pour atteindre le burnup cible de la section d'assemblage. Il est tout de même nécessaire de passer par des points de burnup intermédiaires afin de conserver une bonne finesse dans l'évolution, le franchissement de certains seuils de burnup déclenchant une itération supplémentaire d'autoprotection.

À l'issue de ces calculs de fil on obtient une MICROLIB pour chaque section d'assemblage, dont les concentrations isotopiques correspondent à son burnup exact. L'étape suivante consiste à se servir de cette MICROLIB pour générer une MULTICOMPO à 4 valeurs de bore.

Un calcul de reprise est effectué sur chaque MICROLIBs afin de générer une bibliothèque paramétrée en concentration de bore pour chaque section d'assemblage.

4.2.2 Les calculs de reprise

Chaque calcul de reprise prend en entrée la MICROLIB du calcul de fil correspondant et s'en sert pour générer une MULTICOMPO à 4 valeurs de bore comme décrit au paragraphe 3.1.3 :

- L'équation du transport, sur la géométrie N1 contenant 40 mixtures définies pour 281 groupes d'énergie, est résolue par un calcul de type B avec des conditions aux limites de réflexion (milieu infini). L'inventaire isotopique de ces 40 mixtures est initialisé soit à burnup nul, soit à burnup non nul à partir des résultats du calcul de fil.
- Le flux N1 issu du calcul précédent est utilisé pour condenser les mixtures à 26 groupes d'énergie, que l'on duplique à hauteur de 164 mixtures (stretching pour fitter la géométrie N2).
- L'équation du transport, sur la géométrie N2 contenant 164 mixtures définies pour 26 groupes d'énergie, est résolue par un calcul de type K avec des conditions aux limites de réflexions (milieu infini).
- Le flux N1 issu du calcul précédent est utilisé pour condenser les 164 mixtures à 2 groupe d'énergie **en particularisant certains istopes** et calculer les facteur d'équivalence de Selengut.
- Les 164 mixtures sont alors fusionnées en une seule (dont les isotopes sont fusionnés en

un seul isotope macroscopique résiduel) et celle-ci est agrégée au bon endroit dans la bibliothèque de sections efficaces macroscopiques.

Ce processus est répété 4 fois : une fois pour chaque point de concentration en bore que l'on souahite obtenir dans la bibliothèque macroscopique.

Néanmois on remarquera une différence avec le calcul de reprise décrit au paragraphe 3.1.3 : la particularisation de certains isotopes lors de la condensation à 2 groupes d'énergie (U^{234}, U^{235}, U^{236}, U^{237}, U^{238}, Np^{237}, Np^{238}, Np^{239}, Pu^{238}, Pu^{239}, Pu^{240}, Pu^{241}, Pu^{242}, Cm^{242}, Cm^{243}, Cm^{244}, Cm^{245}, Am^{241}, Am^{242M}, Am^{243}, Pm^{147}, Pm^{148}, Pm^{148M}, Pm^{149}, Sm^{147}, Sm^{148}, Sm^{149}, Sm^{150}, Nd^{146}, Nd^{147}, Nd^{148}, B^{10}, B^{11}, Xe^{135}, I^{135}) au lieu de condenser tous les isotopes (MICR ALL). Ce choix est motivé par le fait que le cluster n'a pas les ressources suffisantes pour supporter une telle charge de cacul.

Alors que le calcul classique DRAGON4 produit une MULTICOMPO à 3360 paramètres, le calcul d'historique produit une MULTICOMPO à 1 paramètre (concentration de bore) pour chaque section assemblage (26 $\times$ le découpage axial).

4.2.3 Le calcul de cœur

La géométrie du réacteur est la même que celle décrite au paragraphe 3.2.4, c'est-à-dire la géométrie réaliste d'un REP à 157 assemblages.

Maintenant que nous avons les bibliothèques de sections macroscopiques MULTICOMPO paramétrées en bore, on cherche la concentration de bore critique de la même façon que dans le calcul de cœur du calcul classique.

Nous en tirons les nouveaux burnups dans chaque section d'assemblage, et il ne reste plus qu'à coupler le calcul de cœur avec les calculs de fils (cf. diagramme 4.2).

4.3 Les modèles

Nous allons étudier et analyser les grandeurs physiques résultant de calculs effectués sous deux conditions différentes : une distribution des différentes grandeurs physiques du réacteur (températures, densités et burnup) homogène en conditions nominales et typique d'une longueur de campagne de REP. Ces deux distributions sont statiques tout au long des calculs.

Le maillage retenu pour le flux est celui du paragraphe 3.2.4, cependant le maillage en températures, densité modérateur et burnup n'est que de 10 mailles axiales (au lieu de 20) à cause d'un bogue (non encore précisemment qualifié) qui limite le calcul des propriétés d'une seule mixture par nœud. Le choix a été fait de poursuivre l'étude avec ce maillage.

Nous comparerons l'évolution de la concentration de bore critique, du facteur de point chaud, des burnups et de la distribution de puissance au cours de la campagne du réacteur tout comme nous l'avons fait au paragraphe 3.2.4. La bibliothèque utilisée est SHEM 281 (Hfaiedh (2006)).

4.3.1 Conditions homogènes

Dans des conditions où les paramètres sont à leur valeur nominale dans tous les assemblages, on s'attend à ne voir aucune différence entre le calcul de cœur et le calcul d'historique, car les paramètres nominaux sont des points de la bibliothèque qui ne nécessitent pas d'interpolation. La calcul avec effet d'historique ne peut donc pas présenter un avantage par rapport au calcul classique dans ce cas particulier.

Résultats

Les résultats sont physiquement équivalents à ceux de la section 3.2.4.

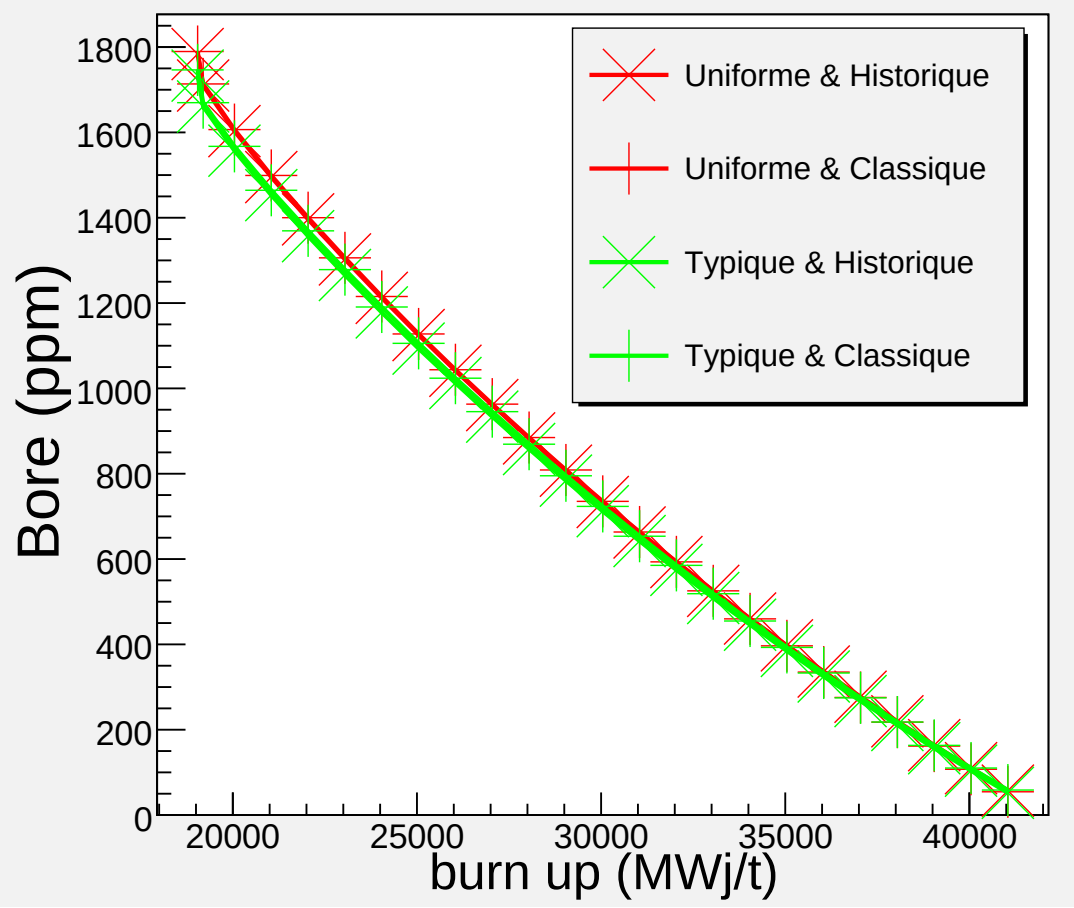
La figure 4.3 représente la concentration de bore critique en ppm (partie pour million) en fonction du burnup moyen du cœur en MWj/t. La figure 4.6 représente le facteur de point chaud (cf. paragraphe 4.3.2) en fonction du burnup en MWj/t. Les courbes rouges correspondent aux calculs avec une distribution uniforme avec des marqueurs « croix droite » pour le calcul classique, « croix oblique » pour le calcul d'historique.

Les tableaux 4.1 et 4.2 récapitulent les valeurs numériques de ces calculs.

Les courbes rouges des figures 4.4, 4.5 et 4.7 reprennent les écarts, tantôt en valeur, tantôt en pourcentage ((historique - classique)/historique).

La figure 4.14 représente la distribution de puissance à la fin de la première itération dans la longueur de campagne.

Figure 4.3 Évolution de la concentration de bore critique



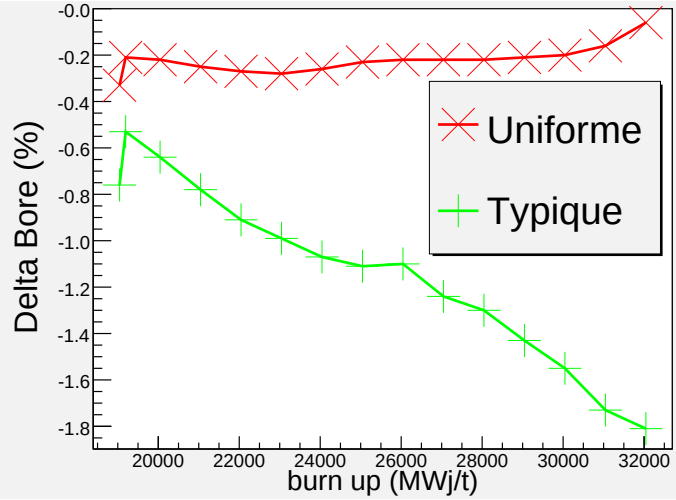


Figure 4.4 Écart de la concentration en pourcentage

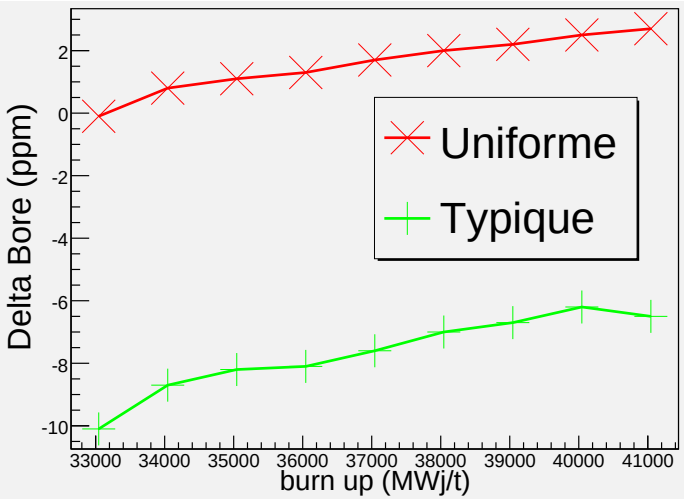


Figure 4.5 Écart de la concentration en ppm

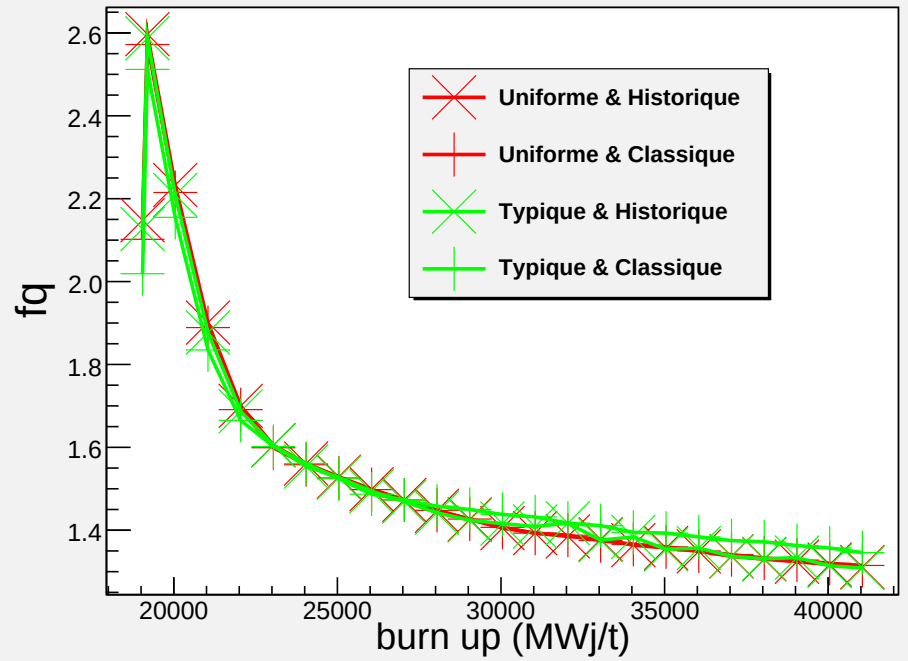


Figure 4.6 Évolution du facteur de point chaud

Analyse

Les résultats sont conformes aux attentes : il n'y a pas d'écart significatif, on observe le même comportement physique. En concentration de bore critique, l'écart est de l'ordre du dixième de pourcent, ce qui nous amène à une erreur de l'ordre de 100 pcm.

Il y aurait parfaite correspondance au démarrage si la carte de burnup initiale (cf. figure 4.8) se situait sur des points de burnup de la bibliothèque. Ce n'est pas le cas et dès le départ le calcul classique interpole la bibliothèque. L'écart ne se creuse pourtant pas au cours la campagne, au fur et à mesure que l'on s'éloigne des points de burnup de la bibliothèque, ce qui est un signe de grande cohérence entre les deux calculs. L'allure de cette courbe sera commenté plus loin (paragraphe).

La distribution de puissance (4.14) est bien uniforme suivant l'axe du cœur.

Nous pouvons conclure que le calcul d'historique est validé par le calcul classique.

4.3.2 Longueur de campagne

Ce calcul a été réalisé dans les mêmes conditions qu'au paragraphe 3.2.4 : Pour cette longueur de campagne typique, on rappelle que chaque assemblage possède un burnup intial homogène parmi {0 ; 13000 ; 26000 ou 39000MWj/t}. Les assemblages sont distribués suivant une carte de burnup qui respecte une symétrie un huitième pour une moyenne d'environ

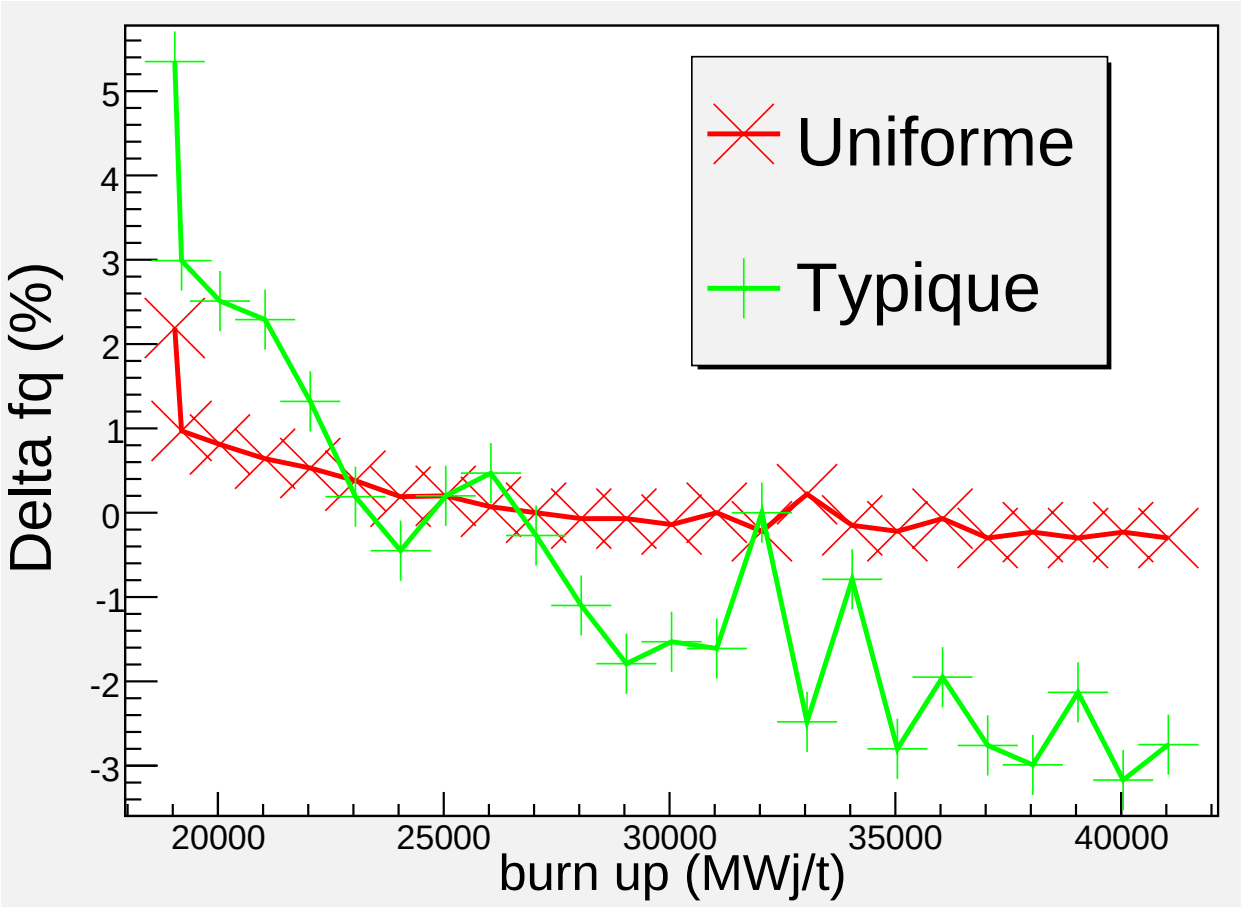


Figure 4.7 Écart du facteur de point chaud en pourcentage

Tableau 4.1 Concentration de bore critique en calcul d'historique et en calcul classique pour une distribution uniforme

	Distribution uniforme			
Burnup	Concentration de bore critique (ppm)			
(MWj/t)	Historique	Classique	Δ	Pourcentage
19045	1783.6	1789.6	-6.0	-0.33 %
19195	1709.9	1713.6	-3.7	-0.21 %
20045	1603.0	1606.6	-3.6	-0.22 %
21045	1495.3	1499.1	-3.8	-0.25 %
22045	1396.2	1400.1	-3.9	-0.27 %
23045	1302.3	1306.0	-3.7	-0.28 %
24045	1212.2	1215.4	-3.2	-0.26 %
25045	1125.0	1127.6	-2.6	-0.23 %
26045	1041.3	1043.7	-2.4	-0.22 %
27045	960.9	963.1	-2.2	-0.22 %
28045	882.8	884.8	-2.0	-0.22 %
29045	807.1	808.8	-1.7	-0.21 %
30045	733.7	735.2	-1.5	-0.20 %
31045	662.3	663.4	-1.1	-0.16 %
32045	593.0	593.4	-0.4	-0.06 %
33045	525.4	525.5	-0.1	-0.01 %
34045	460.6	459.8	0.8	0.17 %
35045	397.7	396.6	1.1	0.27 %
36045	336.4	335.1	1.3	0.38 %
37045	277.3	275.6	1.7	0.61 %
38045	219.7	217.7	2.0	0.91 %
39045	163.8	161.6	2.2	1.36 %
40045	109.8	107.3	2.5	2.32 %
41045	57.2	54.5	2.7	4.95 %

Tableau 4.2 Facteur de point chaud en calcul d'historique et en calcul classique pour une distribution uniforme

	Distribution uniforme			
Burnup	Facteur de point chaud (FQ)			
(MWj/t)	Historique	Classique	Δ	Pourcentage
19045	2.148	2.102	0.046	2.19 %
19195	2.597	2.572	0.025	0.97 %
20045	2.233	2.215	0.018	0.81 %
21045	1.901	1.889	0.012	0.64 %
22045	1.700	1.691	0.009	0.53 %
23045	1.605	1.599	0.006	0.38 %
24045	1.561	1.558	0.003	0.19 %
25045	1.529	1.526	0.003	0.20 %
26045	1.499	1.498	0.001	0.07 %
27045	1.473	1.473	0.000	0.00 %
28045	1.447	1.448	-0.001	-0.07 %
29045	1.426	1.427	-0.001	-0.07 %
30045	1.405	1.407	-0.002	-0.14 %
31045	1.393	1.393	0.000	0.00 %
32045	1.385	1.388	-0.003	-0.22 %
33045	1.378	1.375	0.003	0.22 %
34045	1.364	1.366	-0.002	-0.15 %
35045	1.357	1.360	-0.003	-0.22 %
36045	1.349	1.350	-0.001	-0.07 %
37045	1.337	1.341	-0.004	-0.30 %
38045	1.330	1.333	-0.003	-0.23 %
39045	1.324	1.328	-0.004	-0.30 %
40045	1.317	1.320	-0.003	-0.23 %
41045	1.311	1.315	-0.004	-0.30 %

19000MWj/t au démarrage (cf. figure 4.8).

On reprends la même distribution statique moyenne des valeurs de température et densité que celle choisie au paragraphe 3.2.4, à la différence qu'on en extrait seulement 2 mailles axiales (cf. paragraphe 4.3).

Résultats

La figure 4.3 représente la concentration de bore critique en ppm (partie pour million) en fonction du burnup moyen du cœur en MWj/t. Les courbes vertes correspondent aux calculs avec une distribution typique avec des marqueurs « croix droite » pour le calcul classique, « croix oblique » pour le calcul d'historique.

La figure 4.6 représente le facteur de point chaud tel que défini dans la documentation de DONJON4 (Sekki *et al.* (2009)) en fonction du burnup moyen du cœur. Le facteur de point chaud est le maximum du rapport de la puissance d'une section d'assemblage sur la puissance moyenne du cœur. Etant donné que la modélisation du cœur est faite suivant un maillage cartésien 3D régulier, cette grandeur possède bien une pertinence physique. Les courbes vertes correspondent aux calculs avec une distribution typique avec des marqueurs « croix droite » pour le calcul classique, « croix oblique » pour le calcul d'historique.

Les tableaux 4.3 et 4.4 récapitulent les valeurs numériques de ce calcul.

Les courbes vertes des figures 4.4, 4.5 et 4.7 reprennent les écarts, tantôt en valeur, tantôt en pourcentage ((historique - classique)/historique).

La figure 4.18 représente la distribution de puissance à la fin de la première itération dans la longueur de campagne.

Analyse

En terme de concentration de bore, l'écart se chiffre en plusieurs dizaines de ppm à chaque pas de burnup ce qui n'est pas négligeable. Cet écart se conserve tout au long de la campagne.

En terme de facteur de point chaud, l'écart obtenu oscille de -3% à +3%. On peut observer une augmentation brutale de ce facteur au permier pas d'évolution. Cet artefact n'existe pas dans le calcul classique au paragraphe 3.2.4. Les seules différences entre le calcul classique de la partie précédente et celui-là sont le maillage axial des paramètres qui passe de 20 à 2 et la condensation à 2 groupes du calcul de reprise qui ne fait plus intervenir tous les isopes. Aucune de ces différences ne semblent avoir un lien évident avec l'apparition de cet artefact.

La distribution de puissance (cf. figure 4.18) n'est plus uniforme suivant l'axe du cœur.

L'écart en pourcentage de burnup passe du dixième de pourcent (figure 4.9) au pourcent (figure 4.10). Il y a donc un impact mesurable sur le choix du plan de rechargement du

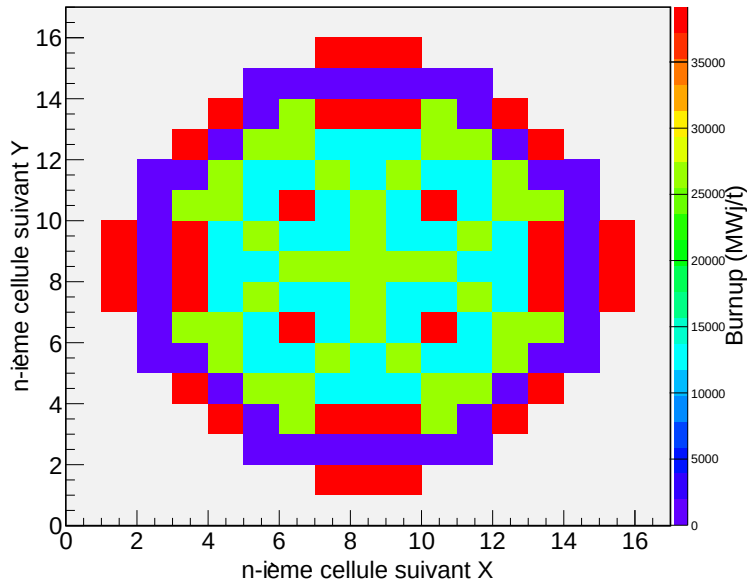


Figure 4.8 Longueur de campagne : carte de burnup intiale

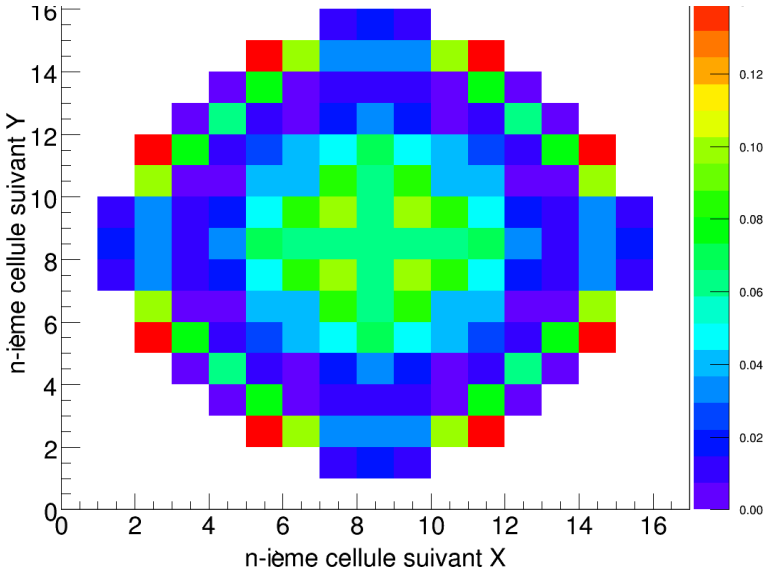


Figure 4.9 Rapport de burnup dans la section centrale (classique-historique)/classique en distribution homogène à la fin de la longueur de campagne

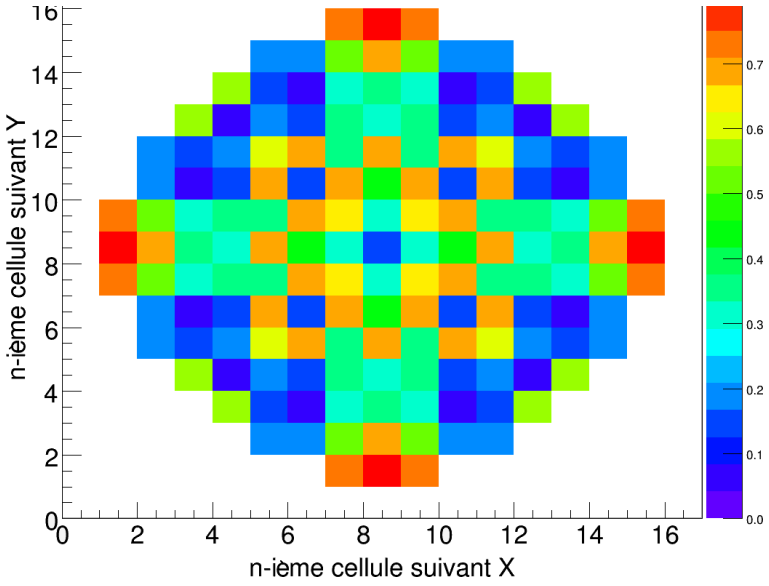


Figure 4.10 Rapport de burnup dans la section centrale (classique-historique)/classique en distribution typique à la fin de la longueur de campagne

Figure 4.11 Section extrême inférieure

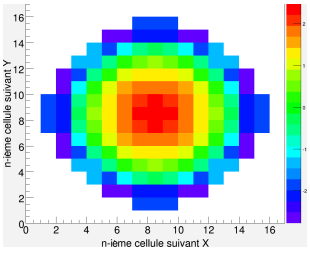


Figure 4.12 Section centrale

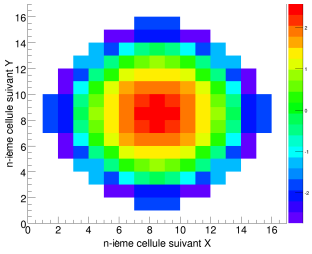


Figure 4.13 Section extrême supérieure

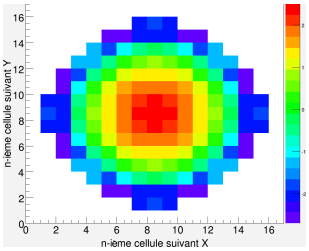


Figure 4.14 Puissance en distribution uniforme après la première itération dans la longueur de campagne

Figure 4.15 Section extrême inférieure

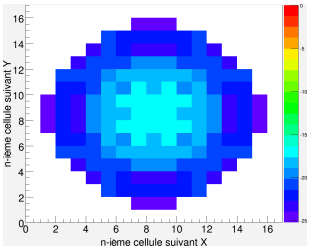


Figure 4.16 Section centrale

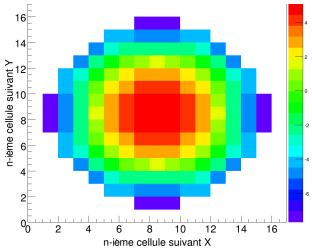


Figure 4.17 Section extrême supérieure

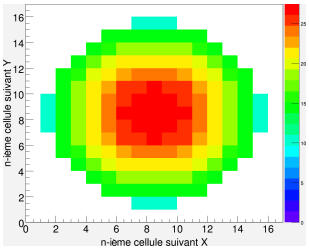


Figure 4.18 Puissance en distribution typique après la première itération dans la longueur de campagne

réacteur à la fin de la longueur de campagne.

4.4 Conclusion

Nous observons un écart sur la concentration de bore critique dans le cas d'une distribution des grandeurs physiques typique d'une longueur de campagne d'un REP d'EDF.

Ces écarts de l'ordre du pourcent apparaissent aussi sur les burnups (figure 4.10) et la distribution de puissance au sein du réacteur. Tout en restant cohérent avec le calcul classique le calcul d'historique ne semble pas garantir un gain de précision important. Une comparaison avec une référence sûre permettrait d'en avoir le cœur net.

4.5 Pour aller plus loin

Il serait intéressant d'étudier et analyser les résultats de calculs effectués sous les conditions d'une distribution fortement hétérogène dont les grandeurs physiques (températures, densités et burnup) sont en plein dehors de points de la bibliothèque. Cela permettrait dans une certaine mesure de majorer le gain de précision que nous apporte le calcul d'historique sur le calcul classique.

Tableau 4.3 Concentration de bore critique en calcul d'historique et en calcul classique pour une distribution typique

	Distribution typique			
Burnup	Concentration de bore critique (ppm)			
(MWj/t)	Historique	Classique	Δ	Pourcentage
19045	1733.2	1746.5	-13.3	-0.76%
19195	1660.8	1669.7	-8.9	-0.53%
20045	1557.2	1567.3	-10.1	-0.64%
21045	1453.0	1464.4	-11.4	-0.78%
22045	1356.8	1369.3	-12.5	-0.91%
23045	1265.8	1278.4	-12.6	-0.99%
24045	1178.1	1190.8	-12.7	-1.07%
25045	1093.3	1105.6	-12.3	-1.11%
26045	1012.6	1023.9	-11.3	-1.10%
27045	933.8	945.5	-11.7	-1.24%
28045	857.9	869.2	-11.3	-1.30%
29045	783.8	795.2	-11.4	-1.43%
30045	712.2	723.4	-11.2	-1.55%
31045	642.2	653.5	-11.3	-1.73%
32045	574.6	585.2	-10.6	-1.81%
33045	508.7	518.8	-10.1	-1.95%
34045	446.1	454.8	-8.7	-1.91%
35045	384.7	392.9	-8.2	-2.09%
36045	324.8	332.9	-8.1	-2.43%
37045	267.1	274.7	-7.6	-2.77%
38045	211.0	218.0	-7.0	-3.21%
39045	156.6	163.3	-6.7	-4.10%
40045	104.0	110.2	-6.2	-5.63%
41045	52.1	58.6	-6.5	-11.09%

Tableau 4.4 Facteur de point chaud en calcul d'historique et en calcul classique pour une distribution typique

	Distribution typique			
Burnup	Facteur de point chaud (FQ)			
(MWj/t)	Historique	Classique	Δ	Pourcentage
19045	2.127	2.019	0.108	5.35%
19195	2.587	2.512	0.075	2.99%
20045	2.209	2.155	0.054	2.51%
21045	1.877	1.835	0.042	2.29%
22045	1.687	1.665	0.022	1.32%
23045	1.605	1.602	0.003	0.19%
24045	1.554	1.561	-0.007	-0.45%
25045	1.527	1.524	0.003	0.20%
26045	1.494	1.487	0.007	0.47%
27045	1.469	1.473	-0.004	-0.27%
28045	1.442	1.458	-0.016	-1.10%
29045	1.424	1.450	-0.026	-1.79%
30045	1.416	1.438	-0.022	-1.53%
31045	1.409	1.432	-0.023	-1.61%
32045	1.417	1.417	0.000	0.00%
33045	1.376	1.411	-0.035	-2.48%
34045	1.384	1.395	-0.011	-0.79%
35045	1.354	1.393	-0.039	-2.80%
36045	1.357	1.384	-0.027	-1.95%
37045	1.337	1.375	-0.038	-2.76%
38045	1.331	1.372	-0.041	-2.99%
39045	1.334	1.363	-0.029	-2.13%
40045	1.314	1.357	-0.043	-3.17%
41045	1.309	1.346	-0.037	-2.75%

CHAPITRE 5

CONCLUSION

Ce travail de mémoire a permis de démontrer d'une part la validité du code DONJON4 comparativement à COCAGNE et d'autre part la viabilité d'un calcul d'historique de REP sur une grappe de serveurs.

5.1 Synthèse des travaux

Les codes DONJON4 et COCAGNE sont cohérents. Nous avons aussi appris que l'interpolation cubique de la bibliothèque de section efficace macroscopique de DONJON4 n'apporte une meilleure précision par rapport à une interpolation linéaire que pour des valeurs faibles de burnup. Cependant le code COCAGNE effectue une correction sur le paramètre xénon, dite « xénon équilibre », qui permet de ne pas ressentir la forte perte de réactivité xénon au démarrage du réacteur (donc à faible valeur de burnup), comme c'est le cas en exploitation. Cette correction rend caduc l'interpolation cubique même à burnup faible.

Les codes DRAGON4 et DONJON4 peuvent être compilés en 64 bits et sont désormais pourvus de fonctionnalités de calcul en parallèle. Celles-ci ont été utilisées sur le cluster Clamart2 d'EDF et un calcul d'historique de REP a pu y être mené. Ce calcul se base sur un couplage entre le code de cœur DONJON4 et le code d'assemblage DRAGON4. Cette alternative à un calcul classique avec bibliothèque est viable mais gourmande en ressource (10 heures sur 33 octoprocesseurs). On peut observer une forte cohérence entre les calculs d'historique et classique en conditions nominales, validant ainsi la solution avec calcul d'historique. Des simplifications d'ordre 1 (maillage plus grossier, abandon de l'inventaire de tous les isotopes, absence de modèle thermique) masquent le gain de précision d'ordre 2 attendu. Néanmoins on observe déjà un écart de l'ordre du 2,5 % sur la concentration en bore dès la première itération dans le calcul de longueur de campagne.

5.2 Limitations de la solution proposée

Le calcul d'historique ne doit être envisagé que s'il apporte un réel gain dans la précision des résultats. Ce gain n'a pas été suffisamment démontré dans ce travail. Des études futures seront nécessaires pour confirmer l'intérêt d'un calcul d'historique.

5.3 Améliorations futures

DONJON4, en tant que code de cœur, manque d'un modèle de contre-réaction thermique. Cette absence le désavantage vis à vis d'un code de cœur concurrent comme COCAGNE, d'autant que la prise en compte des changements de température et de densité du combustible lors de la recherche de la criticité est indispensable pour bénéficier de la finesse d'un calcul avec effet d'historique : dans notre étude les calculs DONJON4 n'ont pu bénéficier que de la distribution statique moyenne de température et de densité issue de COCAGNE.

RÉFÉRENCES

(2004). *Open MPI : Goals, Concept, and Design of a Next Generation MPI Implementation*, Budapest, Hungary.

ASKEW, J. (1972). A characteristics formulation of the neutron transport equation in complicated geometries. Rapport technique AEEW-M 1108, United Kingdom Atomic Energy Establishment, Winfrith.

BELL, G. et GLASSTONE, S. (1970). *Nuclear Reactor Theory*. VanNostrand Reinhold Co., New-York.

GROPP, W., LUSK, E. et SKJELLUM, A. (1996). A high-performance, portable implementation of the mpi message passing interface. *Parallel Computing*.

HFAIEDH, N. (2006). *Nouvelle Méthodologie de Calcul de l'Absorption Résonnante*. Thèse de doctorat, Université Louis Pasteur Strasbourg.

HÉBERT, A. (1987a). Development of the nodal collocation method for solving the neutron diffusion equation. *Ann. nucl. Energy*, 14, 527–541.

HÉBERT, A. (1987b). Trivac, a modular diffusion code for fuel management and design applications. *Nucl. J. of Canada*, 1, 325.

HÉBERT, A. (2006). Towards dragon version4. *Topical Meeting on Advances in Nuclear Analysis and Simulation*.

HÉBERT, A. (2009). *Applied Reactor Physics*. Presses Internationales Polytechnique, Montréal, Québec. ISBN 978-2-553-01436-9.

HÉBERT, A. (2010). Revisiting the ceschino interpolation method. Submitted to the SIAM Journal on Numerical Analysis.

HÉBERT, A. et G., M. (1993). Development of a third-generation superhomogénéisation method for a homogenization of a pressurized water reactor assembly. *Nuclear Science and Engineering*, 115, 129–141.

HÉBERT, A. et MARLEAU, G. (1991). Generalization of the stamm'ler method for the self-shielding of resonant isotopes in arbitrary geometries. *Nuclear Science and Engineering*, 108, 230.

HÉBERT, A., MARLEAU, G. et ROY, R. (1995). Application of the lattice code dragon to candu analysis. *Trans. Am. Nucl. Soc.*

HÉBERT, A., MARLEAU, G. et ROY, R. (2009). A description of the dragon and trivac version4 data structures. Rapport technique IGE-295, Institut de Génie Nucléaire, École Polytechnique de Montréal, Montréal, Québec.

HÉBERT, A. et ROY, R. (1994). A programmer's guide for the gan generalized driver – fortran-77. Rapport technique IGE-158, Institut de Génie Nucléaire, École Polytechnique de Montréal, Montréal, Québec.

KALOS, M. et WHITLOCK, P. (1986). Simulation of stochastique systems : Radiation transport. *Wiley-Interscience*.

LE MER, J. (2007). *Computational schemes on 17x17 PWR assembly (EDF type)*. Mémoire de maîtrise, École Polytechnique de Montréal.

LE TELLIER, R. (2006). *Développement de la méthode des caractéristiques pour le calcul de réseau*. Thèse de doctorat, École Polytechnique de Montréal.

MACFARLANE, R. et MUIR, D. (2000). Code system for producing pointwise and multi-group neutron and photon cross sections from endf/b data. Rapport technique PSR-480/NJOY99, Los Alamos National Laboratory, Los Alamos.

MARLEAU, G., HÉBERT, A. et ROY, R. (2010). A user guide for dragon version4. Rapport technique IGE-294, Institut de Génie Nucléaire, École Polytechnique de Montréal, Montréal, Québec.

PETROVIC, I. et BENOIST, P. (1996). B_N theory : Advances and new models for neutron leakage calculation spatial differencing of the transport equation : Positivity vs. accuracy. *Advance in Nuclear Sciences and Technology*, 24.

REUSS, P. et COSTE-DELCLAUX, M. (2003). Development of computational models used in france for neutron resonance absorption in light water lattices. *Progress in Nuclear Energy*, 42, 237–282.

REYSSET, T. (2009). *Two-level computational schemes for PWR*. Mémoire de maîtrise, École Polytechnique de Montréal.

SANCHEZ, R. et MCCORMIK (1982). A review of neutron transport approximations. *Nuclear Science and Engineering*, 80, 481–535.

SEKKI, D., HÉBERT, A. et CHAMBON, R. (2009). A user guide for donjon version 4. Rapport technique IGE-300, Institut de Génie Nucléaire, École Polytechnique de Montréal, Montréal, Québec.

SELENGUT (1960). Diffusion coefficients for heterogeneous systems. *Transactions of the American Nuclear Society*, 3, 398.

TOUEG, B. (2010). http ://code.google.com/p/dragon-donjon-ascii-viewer/.

VALLERENT, R. (2009). *No title*. Mémoire de maîtrise, École Polytechnique de Montréal.

VAN ROSSUM, G. (1995). Python reference manual. Rapport technique CS-R9525, Communication Web International.

VARIN, E. et MARLEAU, G. (2006). CANDU reactor core simulations using fully coupled DRAGON and DONJON calculations. *Ann. nucl. Energy*, 33, 682–691.

ANNEXE A

Le programme DRAGON-DONJON-ASCIIViewer

Le programme DRAGON DONJON ASCII Viewer est un visualisateur de fichiers ASCII compatibles avec l'ensemble de la distribution DRAGON/DONJON dans leurs versions 3 et 4. Il a été spécifiquement écrit pour faciliter la cherche systématique des sections efficaces dans les structures MULTICOMPO. Le langage de programmation Python muni de la bibliothèque wxPython le rend multiplate-forme : Linux, Windows et Max. Le programme est hébergé sur http ://code.google.com/p/dragon-donjon-ascii-viewer/ et est disponible sous license GPL.

Fonctionnalités La principale fonctionnalité du programme est de pouvoir visualiser et filtrer l'ensemble des calculs élémentaires d'une MULTICOMPO. Pour cela il faut sélectionner dans l'arborescence le nœud CALCULATIONS. Apparaît alors dans la partie droite de la fenêtre un tableau surmonter d'un ensemble de liste choix. Chaque liste de choix correspond à un paramètre de calcul, à l'exception de la dernière qui permet de sélectionner un groupe d'énergie. Il est aussi possible d'effectuer une recherche via le menu Edit (ou le raccourci Ctrl+F sous Windows et Linux). Il est possible de recherche soit parmi les étiquettes soit parmi les valeurs soit les deux. Un appui sur la touche F3 permet d'accéder à l'entrée suivante de la précédente recherche, tandis que Shift+F3 permet d'accéder à l'entrée précédente. Le copier-coller via les raccourcis Ctrl+C/Ctrl+V sont possibles lorsqu'un sélection des cases du tableur est active.

Captures d'écran La figure A.1 représente les première lignes d'une structure MULTICOMPO ouverte avec un éditeur de texte. La figure A.2 représente ce même fichier MULTICOMPO ouvert avec DRAGON DONJON ASCII Viewer. On peut voir que le nœud selectionné dans l'arbre à gauche est le nœud CALCULATIONS, ce qui fait apparaître dans la partie droite la liste des calculs élémentaires triées suivant les paramètres globaux et locaux.

Description des fichiers sources Le code est organisé dans les différents fichiers sources suivant afin de bien séparé les différentes fonctionnalités du programme :

AsciiViewer.py contient l'amorce du programme et constitue l'objet principal qui fait intervenir tous les autres

MyAsciiParser.py contient les fonctions permettant d'ouvrir le fichier ASCII et de parser son contenu

```
04/02/2010                          MultiCompo                                1
->       1      12       3       3                                  <-
SIGNATURE
         4         4         4
L_MULTICOMPO
->       1      12       0      -1                                  <-
MODE
->       2      12       3      20                                  <-
COMMENT
         4         4         4         4         4         4         4         4
         4         4         4         4         4         4         4         4
         4         4         4         4
Multi-parameter reactor database for moderator
->       2      12       0      -1                                  <-
GLOBAL
->       3      12       3       3                                  <-
PARKEY
         4         4         4
MTYPE
->       3      12       3       1                                  <-
PARTYP
         4
VALU
->       3      12       3       2                                  <-
PARFMT
         4         4
STRING
->       3      12       1       2                                  <-
PARCAD
         1         1
->       3      12       1       2                                  <-
PARPAD
         1         1
->       3      12       3       6                                  <-
pval00000001
         4         4         4         4         4         4
CELL18      CELL20
->       3      12       1       1                                  <-
NVALUE
         2
->      -3       0       0       0                                  <-
->       2      12       1      40                                  <-
STATE-VECTOR
         1         2         2        10         1         0         0         0
         0         1         0      2006         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
->       2      12      10       1                                  <-
MIXTURES
->       3       0       0      -1                                  <-   00000001
->       4      12       0      -1                                  <-
TREE
->       5      12       1       2                                  <-
NVP
         3        40
->       5      12       1       1                                  <-
NCALS
         2

                                                file:///home/toueg/Version4_wc/Donjon/data/compo/MultiCompo
```

Figure A.1 Extrait d'une MULTICOMPO au format ASCII

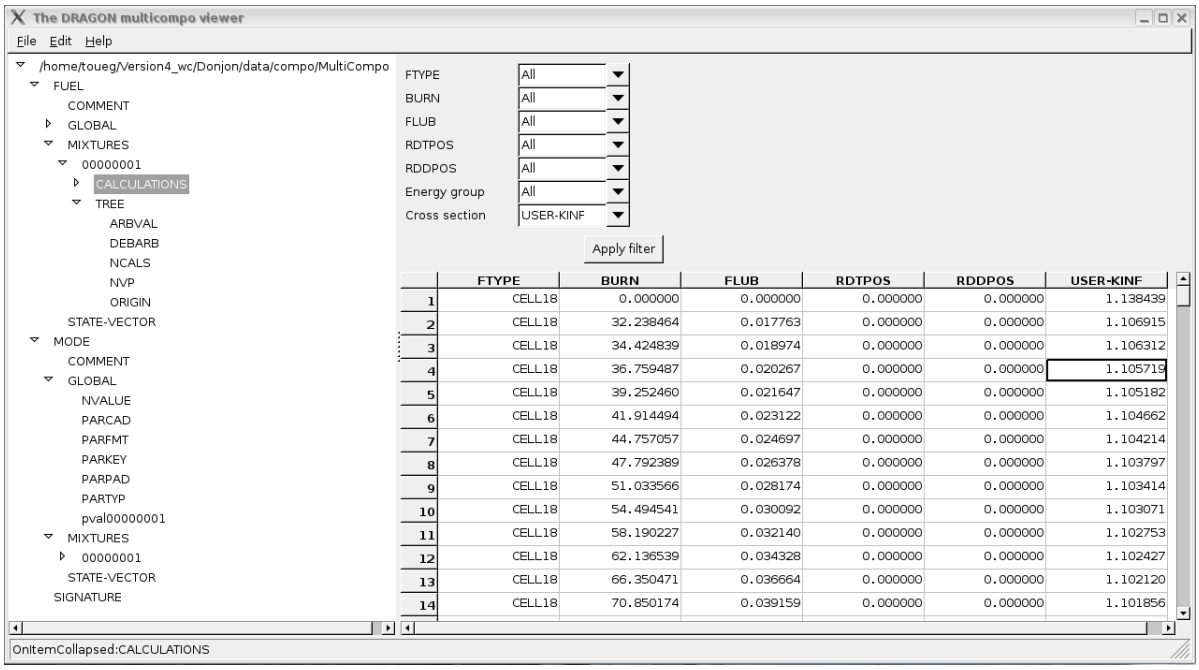


Figure A.2 Capture d'écran du programme DRAGON DONJON ASCII Viewer

MyFindReplaceDialog.py décrit la boîte de dialogue de recherche

MySheet.py contient les fonctions permettant d'afficher le tableur

MyCalculation.py contient les fonctions pour organiser les données MICROLIB d'une MULTICOMPO

MyMenuBar.py décrit le menu en haut de la fenêtre

MyTable.py contient les fonctions permettant de manipuler les données contenues dans le tableur

MyFilterPanel.py décrit les listes de choix pour filtrer le contenu d'une MULTICOMPO

MyRefcase.py permettait de traiter spécifiquement certaines structure EDITION, obsolète dans la version actuelle

MyTreeCtrl.py contient les fonctions permettant de construire et manipler l'arborescence

ANNEXE B

Les modules MPI

DRAGON contains two modules that enables MPI capabilities. These modules are also available to any code built around the GAN generalized driver (Hébert et Roy (1994)), provided that it is properly compiled.

Le module DRVMPI : This module is a utility module related to MPI. It is mostly used to know the rank of the node running the current script. The calling specifications are :

Tableau B.1 – Structure (DRVMPI :)

```
[ NAME := ] DRVMPI: :: [ EDIT iprint ] [ WORLD-SIZE >>ncpu<< ] [ MY-ID
>>rank<< ]
[[ SETLOOP { B0 | B1 } len >>beg<< >>end<< ]]
[ ALLREDUCE { SUM | PROD | MAX | MIN } operand >>result<< ]
[ TIME >>dTime<< ] [ BARRIER ] ;
```

NAME	`character*12` name of a dummy data structure to be possibly used as *NAME2* in the SNDMPI : module. It can be a linked list or an XSM file.
`EDIT`	keyword used to modify the print level *iprint*.
iprint	index used to control the printing of this module. The amount of output produced by this tracking module will vary substantially depending on the print level specified.
`WORLD-SIZE`	keyword used to recover *ncpu*.
ncpu	total number of nodes in the MPI environment.
`MY-ID`	keyword used to recover *rank*.
rank	rank of the node that is running the script
`SETLOOP`	keyword used to partition the set $\{0 \cdots len-1\}$ or $\{1 \cdots len\}$ equitably over the nodes. The result $\{beg \cdots end\}$ is different for each node.

`B0`	keyword used to set the intial position to 0.
`B1`	keyword used to set the intial position to 1.
len	set length.
beg	beginning of the set for the current script. Must be an integer variable.
end	end of the set for the current script. Must be an integer variable.
`ALLREDUCE`	keyword used to make a computation over all the *operand* and to store the result in *result*. Problems can be encountered in the 64-bits version.
`SUM`	keyword used to make `ALLREDUCE` perform a summation.
`PROD`	keyword used to make `ALLREDUCE` perform a multiplication.
`MAX`	keyword used to make `ALLREDUCE` find the maximum over all *operand*.
`MIN`	keyword used to make `ALLREDUCE` find the maximum over all *operand*.
operand	operand in the `ALLREDUCE` calculation. Cannot be a string or a logical value.
result	result of the `ALLREDUCE` calculation. Must be a variable of the same type as *operand*.
`TIME`	keyword used to recover *dTime*.
dTime	time in seconds since an arbitrary time in the past. Must be a double precision variable.
`BARRIER`	keyword used to stop the calculation until every node has reach this barrier.

NAME is always empty. What matters is that *NAME* is no more only declared, it now exists after the call of DRVMPI : module.

The output parameters, denoted as >>*value*<<, are recovered as CLE-2000 variables in the module data located after the `::` keyword.

Le module SNDMPI : This module is used to send or receive a linked list or an XSM file from one node to another one thanks to MPI. It is possible to send a linked list into an XSM file and vice versa. The module is blocked until the message is sent or received. The calling specifications are :

Tableau B.2 – Structure (SNDMPI :)

NAME1 `:= SNDMPI:` *NAME2* `::` [`EDIT` *iprint*] `FROM` *iFrom* `TO` { *iTo* | ALL } ;

NAME1	`character*12` name of the data structure that will be received. It can be a linked list or an XSM file.
NAME2	`character*12` name of the data structure that will be sent. It can be a linked list or an XSM file. Since on the RHS, it has to exist even for receiving scripts. In this case it is recommanded to create an empty data structure *NAME2* by calling the DRVMPI : module.
`EDIT`	keyword used to modify the print level *iprint*.
iprint	index used to control the printing of this module. The amount of output produced by this tracking module will vary substantially depending on the print level specified.
`FROM`	keyword used to set *iFrom*.
iFrom	rank of the node from which *NAME2* has to be read.
`TO`	keyword used to set *iTo*.
iTo	rank of the node where *NAME1* has to be written.
`ALL`	keyword to make every node receive *NAME2* except the node *iFrom*.

ANNEXE C

Script de lancement de DONJON MPI

Ce script permet de lancer DRAGON ou DONJON avec MPI. Il s'emploie de la même manière qu'un script *rdragon* en ajoutant le préfixe *mpiexec [option]*.

```
#!/bin/sh
#!/usr/bin/python
# *-* coding:utf-8 *-*
#
#-----------------------------------------------------------------------
#
#Purpose:
# run DONJON with MPI
#
#Copyright:
# Copyright (C) 2011 Ecole Polytechnique de Montreal
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 2.1 of the License, or (at your option) any later version.
#
#Author(s): B. Toueg
#
#-----------------------------------------------------------------------
#
#
# rdragon/rdonjon/rganlib must be IDENTICAL.
#
# Best thing is to install 'rdonjon'
#          and to link 'ln rdonjon rdragon'
#                      'ln rdonjon rganlib'
#
# author : R. Roy (96-05-03)
# use    : execution of DONJON, DRAGON or GANLIB
# called : { rdonjon | rdragon | rganlib }
#          [ -dDATE ] [ -ePath ] [ -vVERSION ] [ -nNAME ] [ -sSUFF ]
#          [ -pPREF ] [-l] [-T] input [ procs ]
#
# options: you can choose only one of the following:
#           -dDATE    : DATE in format yymmdd for old versions
#           -vVERSION : VERSION number for old versions
#           -nNAME    : NAME of your exec file in $HOME/bin or true name
#                       (default NAME=donjon for rdonjon;
#                                 dragon for rdragon;
#                                 ganlib for rganlib. )
#           -eEXECPath : Execution Path
#                        executable is  $EXECPath/bin/$System/$CODE
```

```
#          you can add as many as you want:
#            procs: can be a directory with CLE-2000 procedures
#                   can be files including CLE-2000 procedures
#
# set -x # uncomment for debugging
#
MPI_IP=$MP_CHILD_INET_ADDR
MPI_NUM=$OMPI_COMM_WORLD_RANK
System=`uname`
MACH=`uname -s`
PROCESSOR=`uname -p`
if [ "$PROCESSOR" = "unknown" ]; then
  PROCESSOR=`uname -m`
fi
HOST=`hostname`
EXECDATE=`date +"%y/%m/%d %H:%M:%S"`
DEBUG="Off"
#DEBUT MODIFICATION B TOUEG
Thisprog=`basename $0`
CODE=Donjon
# if [ $Thisprog = "rdragon4" ]; then
#   CODE=Dragon
# elif [ $Thisprog = "rdonjon4" ]; then
#   CODE=Donjon
# elif [ $Thisprog = "rdonjontest" ]; then
#   CODE=DonjonTest
# elif [ $Thisprog = "rdragontest" ]; then
#   CODE=DragonTest
# else
#   echo "Your script has the following name : " $Thisprog
#   echo "Expected script name: rdragon4, rdonjon4, rdonjontest"
#   exit 1
# fi
#FIN MODIFICATION B TOUEG
COMMAND="$Thisprog $*"
DIRNAME=/home/nucl
# if [ ${HOST#*.} = "recherche.polymtl.ca" ] ; then
#   TMPDIR=/tmp
# elif [ $HOSTNAME = "boltzmann" ] ; then
#   TMPDIR=/tmp
# elif [ $HOSTNAME = "polaris" ] ; then
#   TMPDIR=/tmp
# elif [ $HOSTNAME = "alioth" ] ; then
#   TMPDIR=/tmp
# else
#  TMPDIR=/local00/tmp
TMPDIR=/tmp
# fi
SaveBigFile="yes"
MaxFileSize=2000000
if [ -n "$PBS_JOBID" ]; then
  mkdir $PBS_JOBID
  INPTPath=`pwd`/$PBS_JOBID
else
  DATE=`date +%d%b%Hh%M`
```

```
  mkdir $DATE
  INPTPath=`pwd`/$DATE
fi
LIBRPath=$DIRNAME/lib/libraries$System
PROCPath=$DIRNAME/lib/procs
Inputdir=`pwd`
EXECPath=$HOME/bin/$System/$PROCESSOR
EXEN=$CODE
EXST=$EXEN
LocPath="No"
ExtPath="No"
GPRCPath=$PROCPath/$CODE
UPRCPath=$Inputdir/proc
ALLDIR="$GPRCPath $UPRCPath"
LSSDIR=""
ALLFILE=""
INPUT="No"
OUTNAM=""
NICE="No"
SUFFIX="."$MPI_NUM"."
PREUSR="No"
PRCLIST="No"
TIMER="On"
ACCESS=""
RLIST=""
for param in $*
do
    case $param in
        -g*)    DEBUG="On" ;;
        -d*)    if [ $EXEN = $EXST ]; then
                DATE="`echo $1 | cut -c 3-`"
                EXEN=$CODE$DATE
            else
                echo "The exec file was changed twice."
                exit 1
            fi ;;
        -v*)    if [ $EXEN = $EXST ]; then
                VERSION="`echo $1 | cut -c 3-`"
                EXEN=$CODE$VERSION
            else
                echo "The exec file was changed twice."
                exit 1
            fi ;;
        -s*)    SUFFIX="`echo $1 | cut -c 3-`" ;;
        -k*)    SaveBigFile="yes" ;;
        -l*)    PRCLIST="Yes" ;;
        -n*)    if [ $EXEN = $EXST ]; then
                FXEC=`echo $1 | cut -c 3-`
#               FXEC=${FXEC:-$CODE}
                EXEN=$FXEC
                LocPath="Yes"
            else
                echo "The exec file was changed twice."
                exit 1
            fi ;;
```

```
                -e*)    TMPEPath=`echo $1 | cut -c 3-`
                        ExtPath="Yes"
                TMPEXEC=$TMPEPath/bin/$System ;;
        -p*)    PREUSR="Yes"
            PREFIX="`echo $1 | cut -c 3-`" ;;
        -t*)    TMPDIR="`echo $1 | cut -c 3-`" ;;
        -T*)    TIMER="Off" ;;
        -w*)    OUTNAM="`echo $1 | cut -c 3-`"
            if [ -z "$OUTNAM" ]; then
                NICE="No"
            fi ;;
        -a*)    ACCESS="`echo $1 | cut -c 3-`".access
            if [ ! -f $Inputdir/$ACCESS ]; then
                echo "This access file doesn't exist:" $ACCESS
                exit 1
            fi ;;
        +*)     RLIST="$RLIST -`echo $1 | cut -c 2-`" ;;
        *)  if [ -f $Inputdir/$1 ]; then
                if [ $INPUT = "Yes" ]; then
                    ALLFILE="$ALLFILE $1"
                else
                    MAIN=$1
                    INPUT="Yes"
                fi
            elif [ -f $Inputdir/${1}.x2m ]; then
                if [ $INPUT = "Yes" ] ; then
                    ALLFILE="$ALLFILE ${1}.x2m"
                else
                    MAIN=${1}.x2m
                    INPUT="Yes"
                fi
            elif [ -d $Inputdir/$1 ]; then
                                BEG="`echo $1 | cut -c -3`"
                                BEG2="`echo $1 | cut -c -6`"
                                if [ $BEG = "LSS" -o $BEG2 = "DirLib" ]; then
                                   LSSDIR="$LSSDIR $Inputdir/$1"
                                else
                   ALLDIR="$ALLDIR $Inputdir/$1"
                                fi
            else
                echo "This file is not a proc:" $1
                exit 1
            fi ;;
    esac
    shift
done
EXEC=$EXECPath/$EXEN
#
# check if main input files exist.
if [ $INPUT = "Yes" ]; then
    echo "Main input  :" $MAIN
else
    echo "Main input " $MAIN "not found."
    echo "See" $Thisprog "manual."
    exit 1
```

```
fi
#
# check for access file
MAIN2="` echo $MAIN | cut -d. -f1`"
if [ ! "$ACCESS" ]; then
   if [ -f $Inputdir/$MAIN2.access ]; then
      ACCESS=$MAIN2.access
   fi
fi
#
# check if executable exists.
if [ -x $EXEC ]; then
    echo "Executable  :" $EXEC
else
    echo "Executable  " $EXEC "not found."
    echo "See" $Thisprog "manual."
    exit 1
fi
echo "Proc dir    :" $ALLDIR
#
# create execution directory
#
inum=1
while [ -d $TMPDIR/rundir$MPI_NUM$inum ]
  do
  inum=`expr $inum + 1 `
done
RUNDIR=$TMPDIR/rundir$MPI_NUM$inum
mkdir $RUNDIR
echo "RunDirectory:" $RUNDIR
echo "Submitted   :" $EXECDATE "on" $HOST
#
# determine the name of the main input file
#
NEWINP=p$$
cp $Inputdir/$MAIN $RUNDIR/$NEWINP
#
# get the proc files from directories in the ALLDIR list
#
for DPROC in $ALLDIR
do
    for FullPROC in $DPROC/*
    do
        if [ ! -d $FullPROC -a -f $FullPROC ]; then
#
# if the file is too big, don't copy it, just create a symbolic link
#
            SIZE=`ls -lL $FullPROC | awk '{print $5}'`
            if [ $SIZE -le 3000000 ]; then
                LocPROC="`basename $FullPROC`"
                cp   $DPROC/$LocPROC $RUNDIR/$LocPROC
            else
                LocPROC="`basename $FullPROC`"
                ln -s $DPROC/$LocPROC $RUNDIR/$LocPROC
            fi
```

```
        fi
    done
done
#
# link the self-shielded libraries from directories in the LSSDIR list
#
for DLSS in $LSSDIR
do
    for FullLSS in $DLSS/*
    do
        if [ ! -d $FullLSS -a -f $FullLSS ]; then
            LocLSS="`basename $FullLSS`"
            ln -s $DLSS/$LocLSS $RUNDIR/$LocLSS
        fi
    done
done
#
# get the proc files from files in the ALLFILE list
#
for FullPROC in $ALLFILE
do
    if [ -f $FullPROC ]; then
        LocPROC="`basename $FullPROC`"
        if [ $PREUSR = "Yes" ]; then
            NewPROC=`echo $LocPROC | sed s/$PREFIX//`
        else
            NewPROC=$LocPROC
        fi
        if [ -z "$NewPROC" ]; then
            echo "File" $FullPROC "cannot be copied in" $RUNDIR
            exit 1
        else
            cp $FullPROC $RUNDIR/$NewPROC
        fi
    fi
done
#
# link cross-section libraries for DRAGON
#
if [ $CODE = dragon ]; then
    for FullNAME in $LIBRPath/*
    do
        LocalNAME="`basename $FullNAME`"
        ln -s  $FullNAME $RUNDIR/$LocalNAME
    done
fi
NewEXEC=p$$e
ln -s $EXEC $RUNDIR/$NewEXEC
cd $RUNDIR
#
if [ $ACCESS ]
  then
  $Inputdir/$ACCESS $Inputdir
#else
#  $Inputdird/cea93v4.access $Inputdir
```

```
#  $Inputdird/draglib.access $Inputdir
#  $Inputdird/CANDU_NG.access $Inputdir
fi
#
# replace string of the form xxx in *.c2m *.x2m
#
if [ "$RLIST" ]
   then
   replin $RLIST $NEWINP *.c2m
fi
#
# keep old list of files in the directory
#
ls -l * >p$$o
OUTFIL=p$$pr
echo "**:     EXECUTION SUMMARY" >$OUTFIL
echo "**:"                       >>$OUTFIL
echo "**:CMD " $COMMAND          >>$OUTFIL
echo "**:USR " `whoami`          >>$OUTFIL
echo "**:MAC " `uname -n`        >>$OUTFIL
echo "**:PWD " $INPTPath         >>$OUTFIL
echo "**:EXEC" $EXEC             >>$OUTFIL
echo "**:DATE" `date`            >>$OUTFIL
echo "**:MPI_IP" $MPI_IP         >>$OUTFIL
echo "**:MPI_NUM" $MPI_NUM       >>$OUTFIL
echo "**:"                       >>$OUTFIL
echo "**:HELP man $Thisprog"     >>$OUTFIL
echo "**:"                       >>$OUTFIL
#
# execute the code:
#
touch xxxtime
if [ $DEBUG = "On" ] ; then
  echo "run <"$NEWINP" >>"$OUTFIL > cmd.gdb
#   echo "echo bt\n" >> cmd.gdb
#   echo "bt" >> cmd.gdb
  echo "echo info stack\n" >> cmd.gdb
  echo "info stack" >> cmd.gdb
  echo "quit" >> cmd.gdb
  gdb $RUNDIR"/"$NewEXEC -quiet -x cmd.gdb --batch > gdb.log 2> gdb.err
elif [ $TIMER = "On" ] ; then
  if [ $NICE = "Yes" ]; then
    /usr/bin/time -f "%E real,%U user,%S sys" --output=xxxtime nice $RUNDIR/$NewEXEC <↩
        $NEWINP >>$OUTFIL
  else
    /usr/bin/time -f "%E real,%U user,%S sys" --output=xxxtime $RUNDIR/$NewEXEC <$NEWINP ↩
        >>$OUTFIL
  fi
  cat xxxtime >>$OUTFIL
else
  if [ $NICE = "Yes" ]; then
    nice $RUNDIR/$NewEXEC <$NEWINP >>$OUTFIL
  else
    $RUNDIR/$NewEXEC <$NEWINP >>$OUTFIL
  fi
```

```
fi
RESULT=$MAIN2$SUFFIX"result"
if [ -f $OUTFIL ]; then
    echo "Output file :" $RESULT
    echo "in          :" $INPTPath/
    mv $OUTFIL $INPTPath/$RESULT
fi
cat xxxtime
/bin/rm -f xxxtime
/bin/rm -f $NewEXEC
remainings="nothing"
if [ -f core ]; then
  /bin/rm -f core
  remainings="core"
  SUFFIX="."$MPI_NUM"-"
fi
if [ -f _FIL001 ]; then
  /bin/rm -f _FIL001
  remainings="STAC"
  SUFFIX="."$MPI_NUM"-"
fi
# checking if a proc file has not been properly compiled :
# get proc error flag
PROCERRORFLAG=`grep -c "PREVIOUS PROC (ERROR" $INPTPath/$RESULT`
if [ $PROCERRORFLAG -gt 0 ]; then
  # get all compiled proc
  ERRORCOMPILEDPROC=`grep --binary-files=text "KDRDPR: COMPILING \*" $INPTPath/$RESULT | ↩
      awk '{ print $4 }'`
  # get the last compiled proc
  for i in $ERRORCOMPILEDPROC
  do
    LASTCOMPILEDPROC=$i
  done
fi
# end checking if a proc file has not been properly compiled
# if a proc has not been properly compiled, PROCERRORFLAG > 0
# and the procedure is stored in LASTCOMPILEDPROC so that we can
# bring it back in the INPTPath

nb_of_procs_compiled=`ls *.l2m 2>/dev/null | wc -w`
if [ $nb_of_procs_compiled -gt 0 ]; then
  echo "Creating tar ball of all procs..."
  tar -czvf l2m.tar.gz *.l2m
  echo "Tar ball created."
fi

#
# keep new list of files in the directory
#
if [ $PRCLIST = "No" ] ; then
  touch xxxxx.l2m xxxxx.o2m DLIB
# here we don't keep any proc file (.l2m .o2m) provided they compiled successfuly
  if [ $PROCERRORFLAG -lt 1 ]; then
    NEWLIST=`ls *.l2m *.o2m DLIB`
  else
```

```
# if a proc did not compile it is not matched by grep -v
# so that we can keep it and have a look at it
    NEWLIST=`ls *.l2m *.o2m DLIB | grep -v $LASTCOMPILEDPROC.l2m`
  fi
  for NEWFIL in $NEWLIST
  do
    /bin/rm -f $NEWFIL
  done
fi
ls -l * >p$$n
# modification benjamin toueg ( print $9 -> print $8 )
# only files that are stored in NEWF are subject to be kept
NEWF=`diff -e -b p$$o p$$n | awk '{ print $8 }'`
# fin modification benjamin toueg
KillRunDir="yes"
for NEWFIL in $NEWF
do
    if [ $NEWFIL !=  p$$o ]; then
        Filesize=`du -k $NEWFIL | awk '{ print $1 }'`
        keep="yes"
        if [ $Filesize -ge $MaxFileSize  ]; then
            if [ $SaveBigFile = "no" ]; then
                keep="no"
            fi
        fi
        if [ $keep = "yes" ]; then
            echo "New files    :" $MAIN2$SUFFIX$NEWFIL
            mv $NEWFIL $INPTPath/$MAIN2$SUFFIX$NEWFIL
        else
            KillRunDir="no"
            echo "File         :" $NEWFIL "not transferred"
            echo "Run directory will not be destroyed "
        fi
    fi
done
#
# remove execution directory
#
echo "End of execution on" $HOST
cd $INPTPath
if [ $KillRunDir = "yes" ]; then
  /bin/rm -f -r $RUNDIR
else
  /bin/rm -f  $RUNDIR/p$$o $RUNDIR/p$$n
  echo "Run directory :" $RUNDIR "not destroyed "
fi

#
# if there was no remainings during execution
#
if [ $remainings = "nothing" ]; then
  #
  # printing tail of the proc file that did not compile properly
  #
   if [ $PROCERRORFLAG -gt 0 ]; then
```

```
    TAILTOPRINT=$MAIN2$SUFFIX$LASTCOMPILEDPROC.l2m
    echo "=================================="
    echo "Tail of" $TAILTOPRINT
    echo "=================================="
    if [ -f $TAILTOPRINT ]; then
      tail -n 15 $TAILTOPRINT
    fi
    echo "=================================="
    echo "End of" $TAILTOPRINT
    echo "=================================="
  fi
  #
  # printing tail of result file
  #
  echo "=================================="
  echo "Tail of" $RESULT
  echo "=================================="
  if [ -f $RESULT ]; then
    tail -n 15 $RESULT
  fi
  echo "=================================="
  echo "End of" $RESULT
  echo "=================================="
else
  cat << ERROR
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!                                  !!
!! EXECUTION ERROR: $remainings REMAININGS !!
!!                                  !!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ERROR
fi
```

ANNEXE D

Script de génération des templates CLE-2000

Ce script permet de générer le rendu des templates CLE-2000 avec django 1.3. C'est un fichier exécutable qui prend en entrée le nombre de processeurs (option -*p*) et le découpage axial du cœur (option -*b*).

```
#!/usr/bin/python
# *-* coding:utf-8 *-*
#
#-------------------------------------------------------------------------
#
#Purpose:
# define variables for CLE-2000 templates and render them
#
#Copyright:
# Copyright (C) 2011 Ecole Polytechnique de Montreal
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 2.1 of the License, or (at your option) any later version.
#
#Author(s): B. Toueg
#
#-------------------------------------------------------------------------
#

def usage():
  print """
  -h            print this message
  -p <nproc>    define number of parallel threads
  -b <nbund>    define number of bundle per channel
  -f <filename> file to be processed
"""

import sys,getopt
try:
  opts, args = getopt.getopt(sys.argv[1:], "p:b:s:f:h", ["nproc=","nbund=","slice=","file=↩
      ","help"])
except getopt.GetoptError, err:
  # print help information and exit:
  print str(err) # will print something like "option -a not recognized"
  usage()
  sys.exit(2)
ncpus = 0
n_bund = 20
slice = 0
```

```
template_list = []
for o, a in opts:
  if o in ("-h", "--help"):
    usage()
    sys.exit()
  elif o in ("-p", "--nproc"):
    ncpus = int(a)
  elif o in ("-b", "--nbund"):
    n_bund = int(a)
  elif o in ("-s", "--slice"):
    slice = int(a)
  elif o in ("-f", "--file"):
    template_list.append(a)
  else:
    assert False, "unhandled option"
assert (ncpus != 0), "ncpus must be specified"

if template_list == []:
  template_list = ['hist.template.x2m','./proc/core.template.c2m','./proc/init_fuelmap.↩
      template.c2m', './proc/evo_ass.template.c2m', './proc/evo_ass_depl.template.c2m', '↩
      ./proc/compute_keff.template.c2m']

from django.conf import settings

settings.configure(DEBUG=True, TEMPLATE_DEBUG=True, TEMPLATE_DIRS=("./includes",))

from django.template import Context, Template, loader

# begin context

iter_burn_max = 4
# iter_burn_max = 5

MICR = """
35 U234  U235  U236   U237  U238  Np237 Np238 Np239 Pu238  Pu239
Pu240 Pu241 Pu242  Cm242 Cm243 Cm244 Cm245 Am241 Am242m Am243
Pm147 Pm148 Pm148m Pm149 Sm147 Sm148 Sm149 Sm150 Nd146  Nd147
Nd148 B10   B11    Xe135 I135
"""
# MICR = "ALL"

n_plane = 20

# core height
height = 365.8
bund_height = 365.8 / n_bund
bund_mesh_z = [ "%1.8E"%(45+i*bund_height)  for i in xrange(n_bund+1) ]

bumap="""
39000.0 39000.0 39000.0     0.0     0.0     0.0     0.0     0.0     0.0
    0.0 39000.0     0.0 26000.0 39000.0 39000.0 39000.0 26000.0     0.0
39000.0 39000.0     0.0 26000.0 26000.0 13000.0 13000.0 13000.0 26000.0
26000.0     0.0 39000.0     0.0     0.0 26000.0 13000.0 13000.0 26000.0
13000.0 26000.0 13000.0 13000.0 26000.0     0.0     0.0     0.0 26000.0
26000.0 13000.0 39000.0 13000.0 26000.0 13000.0 39000.0 13000.0 26000.0
```

```
26000.0     0.0 39000.0     0.0 39000.0 13000.0 26000.0 13000.0 13000.0
26000.0 13000.0 13000.0 26000.0 13000.0 39000.0     0.0 39000.0 39000.0
    0.0 39000.0 13000.0 13000.0 26000.0 26000.0 26000.0 26000.0 26000.0
13000.0 13000.0 39000.0     0.0 39000.0 39000.0     0.0 39000.0 13000.0
26000.0 13000.0 13000.0 26000.0 13000.0 13000.0 26000.0 13000.0 39000.0
    0.0 39000.0     0.0 26000.0 26000.0 13000.0 39000.0 13000.0 26000.0
13000.0 39000.0 13000.0 26000.0 26000.0     0.0     0.0     0.0 26000.0
13000.0 13000.0 26000.0 13000.0 26000.0 13000.0 13000.0 26000.0     0.0
    0.0 39000.0     0.0 26000.0 26000.0 13000.0 13000.0 13000.0 26000.0
26000.0     0.0 39000.0 39000.0     0.0 26000.0 39000.0 39000.0 39000.0
26000.0     0.0 39000.0     0.0     0.0     0.0     0.0     0.0     0.0
    0.0 39000.0 39000.0 39000.0
"""
bumap_list = [ '%5.0f.'%float(bu) for bu in bumap.split()]*n_plane
# bumap_list = [ '0.0' for bu in bumap.split()]*n_plane


raw_plane= [
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 3, 4, 5, 6, 5, 4, 3, 0, 0, 0, 0, 0,
0, 0, 0, 0, 7, 8, 9,10,11,10, 9, 8, 7, 0, 0, 0, 0,
0, 0, 0, 7,12,13,14,15,16,15,14,13,12, 7, 0, 0, 0,
0, 0, 3, 8,13,17,18,19,20,19,18,17,13, 8, 3, 0, 0,
0, 0, 4, 9,14,18,21,22,23,22,21,18,14, 9, 4, 0, 0,
0, 1, 5,10,15,19,22,24,25,24,22,19,15,10, 5, 1, 0,
0, 2, 6,11,16,20,23,25,26,25,23,20,16,11, 6, 2, 0,
0, 1, 5,10,15,19,22,24,25,24,22,19,15,10, 5, 1, 0,
0, 0, 4, 9,14,18,21,22,23,22,21,18,14, 9, 4, 0, 0,
0, 0, 3, 8,13,17,18,19,20,19,18,17,13, 8, 3, 0, 0,
0, 0, 0, 7,12,13,14,15,16,15,14,13,12, 7, 0, 0, 0,
0, 0, 0, 0, 7, 8, 9,10,11,10, 9, 8, 7, 0, 0, 0, 0,
0, 0, 0, 0, 0, 3, 4, 5, 6, 5, 4, 3, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
]

mix_list = []
for j in range(n_plane):
  for i in raw_plane:
    if i!=0:
      mix_list.append('%3d'%(i+j*26+3))

file = open("fueltemp")
fueltemp_list = file.read().split()
file.close()
file = open("moderatordensity")
moderatordensity_list = file.read().split()
file.close()

def getPlane(values,n,length=17,default = " "):
  "n is the plane number"
  plane = []
  l = len(values[0])
  line = ""
```

```
  j = n*157
  for i,o in enumerate(raw_plane):
    if i>0 and i%length == 0:
      plane.append(line)
      line = ""
    if o == 0:
      line += " "*(l-1)+default+" "
    else:
      line += values[j]+" "
      j=j+1
  plane.append(line)
  return plane

plane_bundle_slice = reduce(lambda x,y : x+y, [ [nb]*(n_plane/n_bund) for nb in range(↩
    n_bund) ])
fuel_mix_plane_list = [ (getPlane(mix_list,i,17,"1"),getPlane(mix_list,i,12,"1")) for i in↩
     plane_bundle_slice ]
mix_plane_list = [ (getPlane(mix_list,i,17,"0"),getPlane(mix_list,i,12,"0")) for i in ↩
    range(n_bund) ]
fueltemp_plane_list = [ (getPlane(fueltemp_list,i,17),getPlane(fueltemp_list,i,12)) for i ↩
    in range(n_bund) ]
moderatordensity_plane_list = [ (getPlane(moderatordensity_list,i,17),getPlane(↩
    moderatordensity_list,i,10)) for i in range(n_bund) ]
burnup_plane_list = [ (getPlane(bumap_list,i,17),getPlane(bumap_list,i,10)) for i in range↩
    (n_bund) ]


class CAssembly:
  def __init__(self,cpurank,mix_number,fueltemp,moderatordensity,burnup,burninst_rank):
    self.mix = mix_number
    tag = '_%04d'%self.mix
    self.set_tag(tag)
    self.cpurank = cpurank
    self.fueltemp = fueltemp
    self.modtemp = 300.
    self.moderatordensity = moderatordensity
    self.burnup = burnup
    self.butarget = 'butarget%04d'%self.mix
    self.burninst_rank = burninst_rank
  def set_tag(self,tag):
    self.tag = tag
    self.LIB281G = 'LIB281G'+tag
    self.LIBSUBG = 'LIBSUBG'+tag
    self.LIB26G = 'LIB26G'+tag
    self.FLUX1 = 'FLUX1'+tag
    self.FLUX2 = 'FLUX2'+tag
    self.BURN32 = 'BURN32'+tag
    self.BURN156 = 'BURN156'+tag
    self.COMPO = 'COMPO'+tag
    self.XSMCPO = 'XSMCPO'+tag
    self.istep = 'istep'+tag
    self.ibustep = 'ibustep'+tag
    self.iautop = 'iautop'+tag
```

```
fuel_mix = range(4,26*n_bund+4)

assembly_list = []
for rank,fmix in enumerate(fuel_mix):
  try:
    idx = mix_list.index('%3d'%fmix)
  except ValueError:
    idx = -1
  if idx>-1:
    fueltemp = fueltemp_list[idx]
    moderatordensity = moderatordensity_list[idx]
    burnup = bumap_list[idx]
    assembly_list.append( CAssembly((rank % (ncpus-1))+1,fmix,fueltemp,moderatordensity,↩
        burnup,idx) )


if slice > 0:
  assembly_list = assembly_list[:slice]

ppm_list = ["    0.","1800.","1100.","  500."]

# end context

# template rendering

def render_cle2000(filename_input,filename_output,context):
  template = open(filename_input,'r')
  t=Template(template.read())
  c=Context(context)
  print filename_input,'->',filename_output
  output = open(filename_output,'w')
  output.write(t.render(c))

context = {
          "MICR":MICR,
          "fuel_mix":fuel_mix,
          "fuel_mix_plane_list":fuel_mix_plane_list,
          "mix_plane_list":mix_plane_list,
          "fueltemp_list":fueltemp_list,
          "fueltemp_plane_list" : fueltemp_plane_list,
          "moderatordensity_plane_list" : moderatordensity_plane_list,
          "moderatordensity_list":moderatordensity_list,
          "burnup_plane_list":burnup_plane_list,
          "NbRegions": "32",
          "assembly_list": assembly_list,
          "ppm_list":ppm_list,
          "mpi_enabled":True,
          "iter_burn_max":iter_burn_max,
          "xsmSuffixes":['%02d'%i for i in range(1,iter_burn_max+1)],
          "cpumain":0,
          "bund_height":bund_height,
          "bund_mesh_z":bund_mesh_z
          }
for filename_input in template_list:
  render_cle2000(filename_input=filename_input,
```

```
                filename_output=filename_input.replace('.template',''),
                context=context
                )

# fil
my_assembly = CAssembly(cpurank=-1,
                        mix_number=4,
                        fueltemp=550.,
                        moderatordensity=0.717,
                        burnup = 66000.,
                        burninst_rank=-1)
my_assembly.set_tag('')
context.update(
  {"mpi_enabled":False,
   "assembly_list":[my_assembly]
   }
  )
render_cle2000(filename_input='./fil.template.x2m',
               filename_output='./fil.x2m',
               context=context
               )

# reprises
context.update(
  {"mpi_enabled":True,
   "n_reprises":75,
   }
  )
render_cle2000(filename_input='./reprise.template.x2m',
               filename_output='./reprise.x2m',
               context=context
               )

# campaign
mix_list = ["%3d"%4 for i in mix_list]
fuel_mix_plane_list = [ (getPlane(mix_list,i,17,"1"),getPlane(mix_list,i,12,"1")) for i in↩
     plane_bundle_slice ]
mix_plane_list = [ (getPlane(mix_list,i,17,"0"),getPlane(mix_list,i,12,"0")) for i in ↩
    range(n_bund) ]

context.update(
  {"mpi_enabled":False,
   "fuel_mix":[4],
   "fuel_mix_plane_list":fuel_mix_plane_list,
   "mix_plane_list":mix_plane_list
   }
  )
render_cle2000(filename_input='./campaign.template.x2m',
               filename_output='./campaign.x2m',
               context=context
               )
render_cle2000(filename_input='./proc/core.template.c2m',
               filename_output='./proc/core_.c2m',
               context=context
               )
```

```
render_cle2000(filename_input='./proc/init_fuelmap.template.c2m',
               filename_output='./proc/fuelmap_init.c2m',
               context=context
               )

print "template script terminated"
```

ANNEXE E

Templates CLE-2000

Le script CLE-2000 du calcul d'historique :

```
*----------------------------------------------------------------------
*
*Purpose:
* historical 2 level calculation for campaign length calculation
*
*Copyright:
* Copyright (C) 2011 Ecole Polytechnique de Montreal
* This library is free software; you can redistribute it and/or
* modify it under the terms of the GNU Lesser General Public
* License as published by the Free Software Foundation; either
* version 2.1 of the License, or (at your option) any later version.
*
*Author(s): B. Toueg
*
*----------------------------------------------------------------------
*  Contrat EDF/EPM
*  Nom          : main.x2m
*  Type         : fichier DRAGON/DONJON
*  Auteur       : B. Toueg
*  Date         : 2010/04/09
*
MODULE COMPO: EVO: GREP: DELETE: ABORT: END: ;
LINKED_LIST TRACKSS TRACKN1 TRACKN2 ;
LINKED_LIST EDI2G LIB281G_ ;
LINKED_LIST EDIHOM LIBHOM ;
PROCEDURE material Mix_UOX_20 Mix_UOX_32 Autop1 Autop2 ;
PROCEDURE evo_ass evo_ass_depl ;
PROCEDURE Homogenize ConcLIBRARY2 ;

{% if mpi_enabled %}
MODULE DRVMPI: SNDMPI: ;
INTEGER cpumain := 0 ;
INTEGER cpurank totcpu ;
DRVMPI: :: EDIT 0 MY-ID >>cpurank<< WORLD-SIZE >>totcpu<< ;
{% endif %}

XSM_FILE XSM_TRACKSS ;
TRACKSS := XSM_TRACKSS ;
XSM_FILE XSM_TRACKN1 ;
TRACKN1 := XSM_TRACKN1 ;
XSM_FILE XSM_TRACKN2 ;
TRACKN2 := XSM_TRACKN2 ;
```

```
MODULE UTL: ;
LINKED_LIST DONNEES ;
DONNEES := UTL: ::
{% include "data.c2m" %}
;

REAL N_U5_UOX   ;   ! U235 enrichment in UOX-Fuel
REAL N_U5_MOX   ;   ! U235 enrichment in MOX-Fuel
REAL N_Pu8      ;   ! Pu238 percentage in Pu-vector
REAL N_Pu9      ;   ! Pu239 percentage in Pu-vector
REAL N_Pu0      ;   ! Pu240 percentage in Pu-vector
REAL N_Pu1      ;   ! Pu241 percentage in Pu-vector
REAL N_Pu2      ;   ! Pu242 percentage in Pu-vector
REAL N_Am1      ;   ! Am241 percentage in Pu-vector
REAL densU_UOX  ;   ! Theoretical density of Uranium in UOX-Fuel
REAL densU_MOX  ;   ! Theoretical density of Uranium in MOX-Fuel
REAL densPu_MOX ;   ! Theoretical density of Plutonium in MOX-Fuel
REAL teneur1_Pu ;   ! Plutonium content in MOX-high enrichment region
REAL teneur2_Pu ;   ! Plutonium content in MOX-medium enrichment region
REAL teneur3_Pu ;   ! Plutonium content in MOX-low enrichment region
material ::
    >>N_U5_UOX<< >>N_U5_MOX<< >>N_Pu8<< >>N_Pu9<< >>N_Pu0<<
    >>N_Pu1<< >>N_Pu2<< >>N_Am1<< >>densU_UOX<< >>densU_MOX<<
    >>densPu_MOX<< >>teneur1_Pu<< >>teneur2_Pu<< >>teneur3_Pu<< ;
REAL TfuelNomC  ;   ! Nominal Fuel temperature (C)
REAL TModNomC   ;   ! Nominal moderator temperature (C)
REAL dens_mod   ;   ! Nominal moderator density
EVALUATE TfuelNomC TModNomC dens_mod :=
         550.0     300.     0.717  ;
REAL pbore := 500. ;
REAL coeff := 0.199 6.022E+23 1.0E-06 * * 10.8110164 1.0E+24 * / ;
REAL N_B10 ;
*******************************************************************
**************************             ******************************
*************                                    *****************
*                          DONJON INIT
*************                                    *****************
**************************             ******************************
*******************************************************************
PROCEDURE core ;
LINKED_LIST COREGEO ;
INTEGER MaxR ;
COREGEO := core :: >>MaxR<< ;
MODULE USPLIT: ;
LINKED_LIST COREMATEX ;

COREGEO COREMATEX := USPLIT: COREGEO :: NGRP 2 MAXR <<MaxR>>
                     NREFL 3  RMIX 1 2 3
                     NFUEL {{ fuel_mix|length }}  FMIX
{{ fuel_mix|join:' '|wordwrap:70 }}
;
MODULE TRIVAT: ;
LINKED_LIST CORETRACK ;
INTEGER MCFD := 3 ;
CORETRACK := TRIVAT: COREGEO :: MAXR <<MaxR>> MCFD <<MCFD>> ;
```

```
PROCEDURE init_fuelmap ;
LINKED_LIST COREFUELMAP ;
COREFUELMAP COREMATEX := init_fuelmap COREMATEX ;
PROCEDURE get_refl ;
LINKED_LIST MACNOEVO ;
MACNOEVO := get_refl ;
REAL Fuelpwr := 38.425 ; ! W/g ou kW/kg ou MW/t
INTEGER NbRegions := {{ NbRegions }} ; ! 20 or 32
*********************************************************************
*************************             *******************************
*************                                 *****************
*                          LIBRARY CREATION LOOP
*************                                 *****************
*************************             *******************************
*********************************************************************
{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = THEN
EVALUATE TfuelNomC TModNomC dens_mod :=
{{ assembly.fueltemp }}
{{ assembly.modtemp }} {{ assembly.moderatordensity }} ;
LINKED_LIST {{ assembly.LIBSUBG }} ;
{{ assembly.LIBSUBG }} := Mix_UOX_{{ NbRegions }} ::
  <<dens_mod>> <<pbore>> <<N_U5_UOX>> <<densU_UOX>>
  <<TfuelNomC>> <<TModNomC>>
;
LINKED_LIST {{ assembly.LIB281G }} ;
{{ assembly.LIB281G }} Autop1
{{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
ENDIF ;
{% endfor %}

*********************************************************************
*************************             *******************************
*************                                 *****************
*                          DEPLETION
*************                                 *****************
*************************             *******************************
*********************************************************************
REAL BUnextSS ;

{% for assembly in assembly_list %}
REAL Tbeg{{ assembly.tag }} BUbeg{{ assembly.tag }}
     Tend{{ assembly.tag }} BUend{{ assembly.tag }} ;
REAL {{ assembly.butarget }} := {{ assembly.burnup }} ;
INTEGER {{ assembly.istep }}
        {{ assembly.ibustep }}
        {{ assembly.iautop }} := 0 0 0 ;
IF cpurank {{ assembly.cpurank }} = THEN
LINKED_LIST {{ assembly.LIB26G }}
            {{ assembly.FLUX1 }}
            {{ assembly.FLUX2 }}
            {{ assembly.BURN156}}
;
{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
evo_ass_depl
```

```
{{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
'{{ assembly.LIB26G }}'
<<NbRegions>>
<<{{ assembly.istep }}>>
;
EVALUATE {{ assembly.istep }}
         {{ assembly.ibustep }}
         {{ assembly.iautop }} := 1 1 2 ;
EVALUATE Tbeg{{ assembly.tag }} BUbeg{{ assembly.tag }}
         Tend{{ assembly.tag }} BUend{{ assembly.tag }}
         := 0.0 0.0 0.0 0.0 ;
GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>> >>BUnextSS<< ;
ECHO "BUnextSS" BUnextSS ;
{{ assembly.BURN156 }} {{ assembly.LIB26G }} :=
EVO: {{ assembly.LIB26G }} {{ assembly.FLUX2 }} TRACKN2 ::
  EDIT 0 SAVE <<Tbeg{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
;
WHILE BUend{{ assembly.tag }} {{ assembly.butarget }} < DO
  EVALUATE {{ assembly.istep }} := {{ assembly.istep }} 1 + ;
  EVALUATE {{ assembly.ibustep }} := {{ assembly.ibustep }} 1 + ;
*----
*  Depletion - Homogenisation
*----
  EVALUATE BUbeg{{ assembly.tag }} Tbeg{{ assembly.tag }} :=
           BUend{{ assembly.tag }} Tend{{ assembly.tag }} ;
  GREP: DONNEES :: GETVAL 'burn'
                <<{{ assembly.ibustep }}>> >>BUend{{ assembly.tag }}<< ;
  IF BUend{{ assembly.tag }} {{ assembly.butarget }} > THEN
    EVALUATE BUend{{ assembly.tag }} := {{ assembly.butarget }} ;
    EVALUATE {{ assembly.ibustep }} := {{ assembly.ibustep }} 1 - ;
  ENDIF ;
  EVALUATE Tend{{ assembly.tag }} := BUend{{ assembly.tag }} Fuelpwr / ;
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
    EDIT 0
    DEPL <<Tbeg{{ assembly.tag }}>> <<Tend{{ assembly.tag }}>>
    DAY POWR <<Fuelpwr>>
  ;
* 1 group condensation
  EDIHOM Homogenize {{ assembly.FLUX2 }} {{ assembly.LIB26G }}
  TRACKN2 ::  <<NbRegions>> ;
  LIBHOM := EDIHOM :: STEP UP HOMOGENE ;
  {{ assembly.LIB281G }} ConcLIBRARY2
  {{ assembly.LIB281G }} LIBHOM :: <<NbRegions>> ;
  {{ assembly.LIB26G }} EDIHOM LIBHOM := DELETE:
  {{ assembly.LIB26G }} EDIHOM LIBHOM ;

  ECHO "BURNUP_sortie = " BUend{{ assembly.tag }} "Next SS = " BUnextSS ;
  ECHO BUend{{ assembly.tag }} BUnextSS - ABS ;
  IF BUend{{ assembly.tag }} BUnextSS - ABS 1E-5 < THEN
    {{ assembly.LIB281G }} Autop2
    {{ assembly.LIB281G }}
    {{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
    EVALUATE {{ assembly.iautop }} := {{ assembly.iautop }} 1 + ;
```

```
    IF {{ assembly.iautop }} 10 <= THEN
      GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>>
                                      >>BUnextSS<< ;
    ENDIF ;
  ENDIF ;

  {{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  evo_ass_depl          {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  {{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
  '{{ assembly.LIB26G }}'
  <<NbRegions>>
  <<{{ assembly.istep }}>>
  ;
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
    EDIT 0
    SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
  ;
ENDWHILE ;
{{ assembly.LIB26G }} {{ assembly.FLUX2 }}
:= DELETE:
{{ assembly.LIB26G }} {{ assembly.FLUX2 }} ;
ENDIF ; ! cpurank {{ assembly.cpurank }} =
{% endfor %}

*******************************************************************
**************************             ******************************
*************                                    ****************
*                          INITIALISATION
*************                                    ****************
**************************             ******************************
*******************************************************************
REAL cbore0 cbore ;

{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = THEN
LINKED_LIST {{ assembly.COMPO }} ;
EVALUATE TfuelNomC TModNomC dens_mod :=
{{ assembly.fueltemp }}
{{ assembly.modtemp }} {{ assembly.moderatordensity }} ;
{% for ppm in ppm_list %}
************
* {{ ppm }}ppm *
************
EDI2G {{ assembly.LIB26G }} {{ assembly.FLUX2 }} LIB281G_
evo_ass
{{ assembly.LIB281G }} TRACKN1 TRACKN2 ::
'{{ assembly.LIB26G }}'
<<NbRegions>> <<dens_mod>> {{ ppm }} <<TModNomC>> >>cbore0<<
;
LIB281G_ := DELETE: LIB281G_ ;
{% if forloop.first %}
*   ********************
*   * ONCE AND FOR ALL *
```

```
*   ********************
      {{ assembly.COMPO }} := COMPO: {{ assembly.LIB26G }} ::
      EDIT 0
      MAXCAL 5
      COMM 'evolution BURNUP Assembly' ENDC
      PARA 'TCOM' VALU REAL !TEMPERATURE_COMBUSTIBLE
      PARA 'DMOD' VALU REAL !DENSITE_MODERATEUR
      PARA 'CBOR' CONC B10 {{ assembly.LIB26G }} 4 !At_Bore10
      PARA 'NXEN' VALU REAL !NIV_XENON
      PARA 'BURN' VALU REAL !BURNUP
      INIT
      ;
*   ********************
*   * ONCE AND FOR ALL *
*   ********************
{% endif %}
{{ assembly.COMPO }} :=
COMPO:
{{ assembly.COMPO }} EDI2G {{ assembly.BURN156}} {{ assembly.LIB26G }}
::
  EDIT 0
  MACRO
  ! where to recover flux normalization in BURN156
  SET <<Tbeg{{ assembly.tag }}>> DAY
  'TCOM' <<TfuelNomC>>
  'DMOD' <<dens_mod>>
  'NXEN' 1.
  'BURN' <<{{ assembly.butarget }}>>
;
EDI2G := DELETE: EDI2G ;
{% if not forloop.last %}
{{ assembly.LIB26G }} {{ assembly.FLUX2 }}
:= DELETE:
{{ assembly.LIB26G }} {{ assembly.FLUX2 }}
;
{% endif %}
{% endfor %}
ENDIF ;
{% endfor %}

************************************************************************
*************************             ******************************
************                                        ****************
*                          COLLECTING COMPO
************                                        ****************
*************************             ******************************
************************************************************************

{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = cpurank cpumain = + THEN
XSM_FILE {{ assembly.XSMCPO }} ;
IF cpurank cpumain = THEN
LINKED_LIST {{ assembly.COMPO }} ;
{{ assembly.COMPO }} := DRVMPI: :: ;
ENDIF ;
```

```
{{ assembly.XSMCPO }} := SNDMPI: {{ assembly.COMPO }} ::
               EDIT 0 FROM {{ assembly.cpurank }} TO <<cpumain>> ;
ENDIF ;
DRVMPI: :: BARRIER ;
{% endfor %}

*********************************************************************
*************************             *******************************
*************                                     *****************
*                           DONJON CALCULATION
*************                                     *****************
*************************             *******************************
*********************************************************************
REAL Power := 2775. ; ! in MW
REAL fq ;
REAL mean_bu ;
REAL pbore_min pbore_max := 0. 1800. ;
REAL pbore_old ;
REAL keff keff_old ;
REAL eps_r ;
REAL burn_step := 150. ;
LOGICAL cbc_above_10 := $True_L ;
LOGICAL root_found go_brent ;
REAL x1 y1 x2 y2 y3 ;
INTEGER iter_burn iter_cbc ;
MODULE GREP: FIND0: FLPOW: TINST: RESINI: ;
PROCEDURE compute_keff ;
LINKED_LIST COREFLUX POWER BRENT ;

EVALUATE iter_burn := 0 ;
WHILE iter_burn {{ iter_burn_max }} < DO
EVALUATE iter_burn := iter_burn 1 + ;
IF cpurank cpumain = THEN
*----
*  CBC LOOP
*----
IF iter_burn 1 = THEN
  EVALUATE burn_step :=  150. ;
ELSEIF iter_burn 2 = THEN
  EVALUATE burn_step :=  850. ;
ELSE ! iter_burn > 2
  EVALUATE burn_step := 1000. ;
ENDIF ;

GREP: COREFUELMAP :: MEAN 'BURN-INST    ' 1 * >>mean_bu<< ;
EVALUATE keff eps_r := 10. 1E-6 ;
EVALUATE iter_cbc := 0 ;
EVALUATE root_found := $False_L ;
WHILE root_found NOT DO
  EVALUATE iter_cbc := iter_cbc 1 + ;
  IF iter_cbc 1 = THEN
    EVALUATE pbore := pbore_min ;
  ELSEIF iter_cbc 2 = THEN
    EVALUATE pbore := pbore_max ;
  ENDIF ;
```

```
  EVALUATE N_B10 := coeff pbore * ;
  COREFLUX COREMATEX COREFUELMAP := compute_keff
           COREMATEX COREFUELMAP MACNOEVO CORETRACK COREGEO
  :: <<iter_cbc>> <<N_B10>> <<iter_burn>> <<Power>> >>keff<< ;

  EVALUATE root_found := 1. keff - ABS eps_r < ;

  ECHO "pbore       = " pbore ;
  ECHO "K-effective = " keff  ;

  IF root_found NOT THEN
    IF iter_cbc 1 = THEN
      EVALUATE pbore_old := pbore ;
      EVALUATE keff_old := keff ;
    ELSEIF iter_cbc 2 = THEN
      EVALUATE x1 y1 := pbore_old 1. keff_old - ;
      EVALUATE x2 y2 := pbore 1. keff - ;
      EVALUATE go_brent := y1 y2 * 0. < ;
      IF go_brent THEN
        BRENT := FIND0: ::
          POINT X <<x1>> Y <<y1>>
          POINT X <<x2>> Y <<y2>>
          >>root_found<< >>pbore<<
        ;
      ELSE
        EVALUATE root_found := $True_L ;
        EVALUATE pbore keff := 0. 0. ;
      ENDIF ;
    ELSE
      EVALUATE y3 := 1. keff - ;
      BRENT := FIND0: BRENT ::
        Y <<y3>>
        >>root_found<< >>pbore<<
      ;
    ENDIF ;
  ENDIF ;

ENDWHILE ; ! cbc loop

EVALUATE cbc_above_10 := pbore 10. > iter_burn {{ iter_burn_max }} < * ;

IF cbc_above_10 THEN
  POWER := FLPOW: COREFUELMAP COREFLUX CORETRACK COREMATEX ::
    EDIT 1
    PTOT <<Power>>
  ;
  GREP: POWER :: GETVAL 'FORM-BUND    ' 1  >>fq<<  ;
  COREFUELMAP := TINST: COREFUELMAP POWER ::
    EDIT 0
    BURN-STEP <<burn_step>>
  ;
  COREFUELMAP := RESINI: COREFUELMAP :: INST-BVAL SMOOTH ;
  ECHO iter_cbc "iterations in the CBC loop gives :" ;
  ECHO "Average Bu :       " mean_bu " MWd/t" ;
```

```
  ECHO "Burn up increment :" burn_step " MWd/t" ;
  ECHO "CBC :               " pbore "ppm" ;
  ECHO "FQ :                " fq ;
  POWER := DELETE: POWER ;
  IF iter_cbc 2 > THEN
    BRENT := DELETE: BRENT ;
  ENDIF ;
ENDIF ;

{% for suffix in xsmSuffixes %}
IF iter_burn {{ suffix|floatformat }} = THEN
XSM_FILE xsmFMAP{{suffix}} ;
xsmFMAP{{suffix}} := COREFUELMAP ;
XSM_FILE xsmFLUX{{suffix}} ;
xsmFLUX{{suffix}} := COREFLUX ;
ENDIF ;
{% endfor %}

COREFLUX := DELETE: COREFLUX ;

ENDIF ; ! cpurank cpumain =

*****************************************************************
**************************            *******************************
************                                        ****************
*                         SENDING cbc_above_10
************                                        ****************
**************************            *******************************
*****************************************************************

SNDMPI: :: EDIT 0 ITEM <<cbc_above_10>>
                     >>cbc_above_10<<
                FROM <<cpumain>> TO ALL ;

IF cbc_above_10 NOT THEN
END: ;
ENDIF ;

*****************************************************************
**************************            *******************************
************                                        ****************
*                         SENDING BUTNUP TARGET
************                                        ****************
**************************            *******************************
*****************************************************************

{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = cpurank cpumain = + THEN
IF cpurank cpumain = THEN
GREP: COREFUELMAP :: GETVAL 'BURN-INST    '
                            {{ assembly.burninst_rank|add:'1' }}
                            >>{{ assembly.butarget }}<< ;
ENDIF ;
ECHO "{{ assembly.butarget }} " {{ assembly.butarget }} ;
SNDMPI: :: EDIT 0 ITEM <<{{ assembly.butarget }}>>
```

```
                    >>{{ assembly.butarget }}<<
                FROM <<cpumain>> TO {{ assembly.cpurank }} ;
ECHO "{{ assembly.butarget }} " {{ assembly.butarget }} ;
ENDIF ;
DRVMPI: :: BARRIER ;
{% endfor %}

************************************************************************
*************************             ******************************
************                                      ****************
*                          DEPLETION
************                                      ****************
*************************             ******************************
************************************************************************

{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = THEN
WHILE BUend{{ assembly.tag }} {{ assembly.butarget }} < DO
  EVALUATE {{ assembly.istep }} := {{ assembly.istep }} 1 + ;
  EVALUATE {{ assembly.ibustep }} := {{ assembly.ibustep }} 1 + ;
*----
*  Depletion - Homogenisation
*----
  EVALUATE BUbeg{{ assembly.tag }} Tbeg{{ assembly.tag }} :=
           BUend{{ assembly.tag }} Tend{{ assembly.tag }} ;
  GREP: DONNEES :: GETVAL 'burn'
               <<{{ assembly.ibustep }}>> >>BUend{{ assembly.tag }}<< ;
  IF BUend{{ assembly.tag }} {{ assembly.butarget }} > THEN
    EVALUATE BUend{{ assembly.tag }} := {{ assembly.butarget }} ;
    EVALUATE {{ assembly.ibustep }} := {{ assembly.ibustep }} 1 - ;
  ENDIF ;
  EVALUATE Tend{{ assembly.tag }} := BUend{{ assembly.tag }} Fuelpwr / ;
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
    EDIT 0
    DEPL <<Tbeg{{ assembly.tag }}>> <<Tend{{ assembly.tag }}>>
    DAY POWR <<Fuelpwr>>
  ;
* 1 group condensation
  EDIHOM Homogenize {{ assembly.FLUX2 }} {{ assembly.LIB26G }}
  TRACKN2 ::  <<NbRegions>> ;
  LIBHOM := EDIHOM :: STEP UP HOMOGENE ;
  {{ assembly.LIB281G }} ConcLIBRARY2
  {{ assembly.LIB281G }} LIBHOM :: <<NbRegions>> ;
  {{ assembly.LIB26G }} EDIHOM LIBHOM := DELETE:
  {{ assembly.LIB26G }} EDIHOM LIBHOM ;

  ECHO "BURNUP_sortie = " BUend{{ assembly.tag }} "Next SS = " BUnextSS ;
  ECHO BUend{{ assembly.tag }} BUnextSS - ABS ;
  IF BUend{{ assembly.tag }} BUnextSS - ABS 1E-5 < THEN
    {{ assembly.LIB281G }} Autop2
    {{ assembly.LIB281G }}
    {{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
    EVALUATE {{ assembly.iautop }} := {{ assembly.iautop }} 1 + ;
```

```
    IF {{ assembly.iautop }} 10 <= THEN
      GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>>
                                      >>BUnextSS<< ;
    ENDIF ;
  ENDIF ;

  {{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  evo_ass_depl          {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  {{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
  '{{ assembly.LIB26G }}'
  <<NbRegions>>
  <<{{ assembly.istep }}>>
  ;
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
    EDIT 0
    SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
  ;
ENDWHILE ;
{{ assembly.LIB26G }} {{ assembly.FLUX2 }}
:= DELETE:
{{ assembly.LIB26G }} {{ assembly.FLUX2 }} ;
ENDIF ; ! cpurank {{ assembly.cpurank }} =
{% endfor %}

*******************************************************************************
***************************             *******************************
*************                                       ****************
*                          INITIALISATION
*************                                       ****************
***************************             *******************************
*******************************************************************************

{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = THEN
{{ assembly.COMPO }} := DELETE:
{{ assembly.COMPO }} ;
EVALUATE TfuelNomC TModNomC dens_mod :=
{{ assembly.fueltemp }}
{{ assembly.modtemp }} {{ assembly.moderatordensity }} ;
{% for ppm in ppm_list %}
************
* {{ ppm }}ppm *
************
EDI2G {{ assembly.LIB26G }} {{ assembly.FLUX2 }} LIB281G_
evo_ass
{{ assembly.LIB281G }} TRACKN1 TRACKN2 ::
'{{ assembly.LIB26G }}'
<<NbRegions>> <<dens_mod>> {{ ppm }} <<TModNomC>> >>cbore0<<
;
LIB281G_ := DELETE: LIB281G_ ;
{% if forloop.first %}
*   ********************
*   * ONCE AND FOR ALL *
```

```
*   ********************
      {{ assembly.COMPO }} := COMPO: {{ assembly.LIB26G }} ::
      EDIT 0
      MAXCAL 5
      COMM 'evolution BURNUP Assembly' ENDC
      PARA 'TCOM' VALU REAL !TEMPERATURE_COMBUSTIBLE
      PARA 'DMOD' VALU REAL !DENSITE_MODERATEUR
      PARA 'CBOR' CONC B10 {{ assembly.LIB26G }} 4 !At_Bore10
      PARA 'NXEN' VALU REAL !NIV_XENON
      PARA 'BURN' VALU REAL !BURNUP
      INIT
      ;
*   ********************
*   * ONCE AND FOR ALL *
*   ********************
{% endif %}
{{ assembly.COMPO }} :=
COMPO:
{{ assembly.COMPO }} EDI2G {{ assembly.BURN156}} {{ assembly.LIB26G }}
::
  EDIT 0
  MACRO
  ! where to recover flux normalization in BURN156
  SET <<Tbeg{{ assembly.tag }}>> DAY
  'TCOM' <<TfuelNomC>>
  'DMOD' <<dens_mod>>
  'NXEN' 1.
  'BURN' <<{{ assembly.butarget }}>>
;
EDI2G := DELETE: EDI2G ;
{% if not forloop.last %}
{{ assembly.LIB26G }} {{ assembly.FLUX2 }}
:= DELETE:
{{ assembly.LIB26G }} {{ assembly.FLUX2 }}
;
{% endif %}
{% endfor %}
ENDIF ;
{% endfor %}

************************************************************************
*************************             ******************************
*************                                        ****************
*                          COLLECTING COMPO
*************                                        ****************
*************************             ******************************
************************************************************************

{% for assembly in assembly_list %}
IF cpurank {{ assembly.cpurank }} = cpurank cpumain = + THEN
{{ assembly.XSMCPO }} DELETE: {{ assembly.XSMCPO }} ;
{{ assembly.XSMCPO }} := SNDMPI: {{ assembly.COMPO }} ::
               EDIT 0 FROM {{ assembly.cpurank }} TO <<cpumain>> ;
ENDIF ;
DRVMPI: :: BARRIER ;
```

```
{% endfor %}

ENDWHILE ;

END: ;
QUIT .
```

Le script CLE-2000 du calcul de fil :

```
*----------------------------------------------------------------------
*
*Purpose:
* 2 level calculation to determine concentration
* for reuse when building MULTICOMPO
*
*Copyright:
* Copyright (C) 2011 Ecole Polytechnique de Montreal
* This library is free software; you can redistribute it and/or
* modify it under the terms of the GNU Lesser General Public
* License as published by the Free Software Foundation; either
* version 2.1 of the License, or (at your option) any later version.
*
*Author(s): B. Toueg
*
*----------------------------------------------------------------------
*  Contrat EDF/EPM
*  Nom          : fil.template.x2m
*  Type         : fichier DRAGON
*  Auteur       : R. Letellier, R. Vallerent, B. Toueg
*  Date         : 2009/09/16
*
*-----
*  Define STRUCTURES and MODULES used
*-----
MODULE LIB: EDI: DELETE: GREP: END: EVO: UTL: COMPO: ABORT:            ;
LINKED_LIST
  TRACKSS TRACKN1 TRACKN2
  LIBRARY LIBHOM
  BURN32 BURN156
  EDIHOM
  COMPO
;
PROCEDURE material Mix_UOX_20 Mix_UOX_32 Autop1 Autop2 evo_ass_depl ;
PROCEDURE evo_ass Homogenize ConcLIBRARY2 ;

XSM_FILE XSM_TRACKSS ;
TRACKSS := XSM_TRACKSS ;
XSM_FILE XSM_TRACKN1 ;
TRACKN1 := XSM_TRACKN1 ;
XSM_FILE XSM_TRACKN2 ;
TRACKN2 := XSM_TRACKN2 ;

MODULE UTL: ;
```

```
LINKED_LIST DONNEES ;
DONNEES := UTL: ::
{% include "data.c2m" %}
;

REAL N_U5_UOX   ;   ! U235 enrichment in UOX-Fuel
REAL N_U5_MOX   ;   ! U235 enrichment in MOX-Fuel
REAL N_Pu8      ;   ! Pu238 percentage in Pu-vector
REAL N_Pu9      ;   ! Pu239 percentage in Pu-vector
REAL N_Pu0      ;   ! Pu240 percentage in Pu-vector
REAL N_Pu1      ;   ! Pu241 percentage in Pu-vector
REAL N_Pu2      ;   ! Pu242 percentage in Pu-vector
REAL N_Am1      ;   ! Am241 percentage in Pu-vector
REAL densU_UOX  ;   ! Theoretical density of Uranium in UOX-Fuel
REAL densU_MOX  ;   ! Theoretical density of Uranium in MOX-Fuel
REAL densPu_MOX ;   ! Theoretical density of Plutonium in MOX-Fuel
REAL teneur1_Pu ;   ! Plutonium content in MOX-high enrichment region
REAL teneur2_Pu ;   ! Plutonium content in MOX-medium enrichment region
REAL teneur3_Pu ;   ! Plutonium content in MOX-low enrichment region
material ::
    >>N_U5_UOX<< >>N_U5_MOX<< >>N_Pu8<< >>N_Pu9<< >>N_Pu0<<
    >>N_Pu1<< >>N_Pu2<< >>N_Am1<< >>densU_UOX<< >>densU_MOX<<
    >>densPu_MOX<< >>teneur1_Pu<< >>teneur2_Pu<< >>teneur3_Pu<< ;

REAL TfuelNomC  ;   ! Nominal Fuel temperature (C)
REAL TModNomC   ;   ! Nominal moderator temperature (C)
REAL dens_mod   ;   ! Nominal moderator density
EVALUATE TfuelNomC TModNomC dens_mod :=
         550.0     300.     0.717  ;
REAL pbore := 500. ;
REAL coeff := 0.199 6.022E+23 1.0E-06 * * 10.8110164 1.0E+24 * / ;
REAL N_B10 ;
REAL Fuelpwr := 38.425 ; ! W/g ou kW/kg ou MW/t

*----
*  Calculation options
*----
INTEGER NbRegions := {{ NbRegions }} ; ! 20 or 32
STRING Library := "DLIBJ3_281" ; ! CEA05V4_281 or DLIBJ3_281
************************************************************************
**************************             *******************************
*************                                      ****************
*                          LIBRARY CREATION LOOP
*************                                      ****************
**************************             *******************************
************************************************************************
{% for assembly in assembly_list %}
EVALUATE TfuelNomC TModNomC dens_mod :=
{{ assembly.fueltemp }}
{{ assembly.modtemp }} {{ assembly.moderatordensity }} ;
LINKED_LIST {{ assembly.LIBSUBG }} ;
{{ assembly.LIBSUBG }} := Mix_UOX_{{ NbRegions }} ::
  <<dens_mod>> <<pbore>> <<N_U5_UOX>> <<densU_UOX>>
  <<TfuelNomC>> <<TModNomC>>
;
```

```
LINKED_LIST {{ assembly.LIB281G }} ;
{{ assembly.LIB281G }} Autop1
{{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
{% endfor %}

*********************************************************************
*************************             *******************************
************                                        ****************
*                          DEPLETION
************                                        ****************
*************************             *******************************
*********************************************************************
REAL BUnextSS ;

{% for assembly in assembly_list %}
REAL Tbeg{{ assembly.tag }} BUbeg{{ assembly.tag }}
     Tend{{ assembly.tag }} BUend{{ assembly.tag }} ;
REAL {{ assembly.butarget }} := {{ assembly.burnup }} ;
INTEGER {{ assembly.istep }}
        {{ assembly.ibustep }}
        {{ assembly.iautop }} := 0 0 0 ;
LINKED_LIST {{ assembly.LIB26G }}
            {{ assembly.FLUX1 }}
            {{ assembly.FLUX2 }}
            {{ assembly.BURN156}}
;
{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
evo_ass_depl
{{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
'{{ assembly.LIB26G }}'
<<NbRegions>>
<<{{ assembly.istep }}>>
;
EVALUATE {{ assembly.istep }}
         {{ assembly.ibustep }}
         {{ assembly.iautop }} := 1 1 2 ;
EVALUATE Tbeg{{ assembly.tag }} BUbeg{{ assembly.tag }}
         Tend{{ assembly.tag }} BUend{{ assembly.tag }}
         := 0.0 0.0 0.0 0.0 ;
GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>> >>BUnextSS<< ;
ECHO "BUnextSS" BUnextSS ;
{{ assembly.BURN32 }} {{ assembly.LIB281G }} :=
EVO: {{ assembly.LIB281G }} {{ assembly.FLUX1 }} TRACKN1 ::
  EDIT 0 SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
;
{{ assembly.BURN156 }} {{ assembly.LIB26G }} :=
EVO: {{ assembly.LIB26G }} {{ assembly.FLUX2 }} TRACKN2 ::
  EDIT 0 SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
;
WHILE BUend{{ assembly.tag }} {{ assembly.butarget }} < DO
  EVALUATE {{ assembly.istep }} := {{ assembly.istep }} 1 + ;
  EVALUATE {{ assembly.ibustep }} := {{ assembly.ibustep }} 1 + ;
*----
*  Depletion - Homogenisation
*----
```

```
  EVALUATE BUbeg{{ assembly.tag }} Tbeg{{ assembly.tag }} :=
           BUend{{ assembly.tag }} Tend{{ assembly.tag }} ;
  GREP: DONNEES :: GETVAL 'burn'
                <<{{ assembly.ibustep }}>> >>BUend{{ assembly.tag }}<< ;
  IF BUend{{ assembly.tag }} {{ assembly.butarget }} > THEN
    EVALUATE BUend{{ assembly.tag }} := {{ assembly.butarget }} ;
    EVALUATE {{ assembly.ibustep }} := {{ assembly.ibustep }} 1 - ;
  ENDIF ;
  EVALUATE Tend{{ assembly.tag }} := BUend{{ assembly.tag }} Fuelpwr / ;
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
    EDIT 0
    DEPL <<Tbeg{{ assembly.tag }}>> <<Tend{{ assembly.tag }}>>
    DAY POWR <<Fuelpwr>>
  ;
* 1 group condensation
  EDIHOM Homogenize {{ assembly.FLUX2 }} {{ assembly.LIB26G }}
  TRACKN2 ::  <<NbRegions>> ;
  LIBHOM := EDIHOM :: STEP UP HOMOGENE ;
  {{ assembly.LIB281G }} ConcLIBRARY2
  {{ assembly.LIB281G }} LIBHOM :: <<NbRegions>> ;
  {{ assembly.LIB26G }} EDIHOM LIBHOM := DELETE:
  {{ assembly.LIB26G }} EDIHOM LIBHOM ;

  ECHO "BURNUP_sortie = " BUend{{ assembly.tag }} "Next SS = " BUnextSS ;
  ECHO BUend{{ assembly.tag }} BUnextSS - ABS ;
  IF BUend{{ assembly.tag }} BUnextSS - ABS 1E-5 < THEN
    {{ assembly.LIB281G }} Autop2
    {{ assembly.LIB281G }}
    {{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
    EVALUATE {{ assembly.iautop }} := {{ assembly.iautop }} 1 + ;
    IF {{ assembly.iautop }} 10 <= THEN
      GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>>
                                      >>BUnextSS<< ;
    ENDIF ;
  ENDIF ;

  {{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  evo_ass_depl          {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  {{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
  '{{ assembly.LIB26G }}'
  <<NbRegions>>
  <<{{ assembly.istep }}>>
  ;

  {{ assembly.BURN32 }} {{ assembly.LIB281G }} := EVO:
  {{ assembly.BURN32 }} {{ assembly.LIB281G }}
  {{ assembly.FLUX1 }} TRACKN1 ::
    EDIT 0
    SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
  ;
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
```

```
    EDIT 0
    SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
  ;
ENDWHILE ;

XSM_FILE BURN156S ;
BURN156S := {{ assembly.BURN156 }} ;
XSM_FILE BURN32S ;
BURN32S  := {{ assembly.BURN32 }} ;
XSM_FILE LIBEQS ;
LIBEQS := {{ assembly.LIB26G }} ;

{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
:= DELETE:
{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }} ;
{% endfor %}

END: ;
QUIT .
```

Le script CLE-2000 du calcul de reprise :

```
*----------------------------------------------------------------------
*
*Purpose:
* 2 level calculation to build MULTICOMPO
*
*Copyright:
* Copyright (C) 2011 Ecole Polytechnique de Montreal
* This library is free software; you can redistribute it and/or
* modify it under the terms of the GNU Lesser General Public
* License as published by the Free Software Foundation; either
* version 2.1 of the License, or (at your option) any later version.
*
*Author(s): B. Toueg
*
*----------------------------------------------------------------------
*  Contrat EDF/EPM
*  Nom          : reprise.template.x2m
*  Type         : fichier DRAGON
*  Auteur       : R. Vallerent, B. Toueg
*  Date         : 2009/09/23
*
*-----
*  Define STRUCTURES and MODULES used
*-----
MODULE LIB: EDI: DELETE: GREP: END: EVO: UTL: COMPO: ABORT:           ;
LINKED_LIST
  TRACKSS TRACKN1 TRACKN2
  LIBRARY
  BURN32 BURN156
  EDI2G
  COMPO
```

```
;
PROCEDURE material Mix_UOX_20 Mix_UOX_32 Autop1 Autop2 evo_ass_depl ;
PROCEDURE FillLib Cond_2gr_SPH ;

XSM_FILE XSM_TRACKSS ;
TRACKSS := XSM_TRACKSS ;
XSM_FILE XSM_TRACKN1 ;
TRACKN1 := XSM_TRACKN1 ;
XSM_FILE XSM_TRACKN2 ;
TRACKN2 := XSM_TRACKN2 ;

MODULE UTL: ;
LINKED_LIST DONNEES ;
DONNEES := UTL: ::
{% include "data.c2m" %}
;

REAL N_U5_UOX   ;   ! U235 enrichment in UOX-Fuel
REAL N_U5_MOX   ;   ! U235 enrichment in MOX-Fuel
REAL N_Pu8      ;   ! Pu238 percentage in Pu-vector
REAL N_Pu9      ;   ! Pu239 percentage in Pu-vector
REAL N_Pu0      ;   ! Pu240 percentage in Pu-vector
REAL N_Pu1      ;   ! Pu241 percentage in Pu-vector
REAL N_Pu2      ;   ! Pu242 percentage in Pu-vector
REAL N_Am1      ;   ! Am241 percentage in Pu-vector
REAL densU_UOX  ;   ! Theoretical density of Uranium in UOX-Fuel
REAL densU_MOX  ;   ! Theoretical density of Uranium in MOX-Fuel
REAL densPu_MOX ;   ! Theoretical density of Plutonium in MOX-Fuel
REAL teneur1_Pu ;   ! Plutonium content in MOX-high enrichment region
REAL teneur2_Pu ;   ! Plutonium content in MOX-medium enrichment region
REAL teneur3_Pu ;   ! Plutonium content in MOX-low enrichment region
material ::
    >>N_U5_UOX<< >>N_U5_MOX<< >>N_Pu8<< >>N_Pu9<< >>N_Pu0<<
    >>N_Pu1<< >>N_Pu2<< >>N_Am1<< >>densU_UOX<< >>densU_MOX<<
    >>densPu_MOX<< >>teneur1_Pu<< >>teneur2_Pu<< >>teneur3_Pu<< ;
REAL TfuelNomC  ;   ! Nominal Fuel temperature (C)
REAL TModNomC   ;   ! Nominal moderator temperature (C)
REAL dens_mod   ;   ! Nominal moderator density
EVALUATE TfuelNomC TModNomC dens_mod :=
         550.0     300.    0.717  ;
REAL pbore := 500. ;
REAL coeff := 0.199 6.022E+23 1.0E-06 * * 10.8110164 1.0E+24 * / ;
REAL N_B10 ;
REAL Fuelpwr := 38.425 ; ! W/g ou kW/kg ou MW/t

*----
*  Calculation options
*----
INTEGER NbRegions := {{ NbRegions }} ; ! 20 or 32
STRING Library := "DLIBJ3_281" ; ! CEA05V4_281 or DLIBJ3_281

REAL n_xenon := 1. ; ! 0.0 or 1.0
REAL TfuelC  ;   ! Fuel temperature (C)
REAL TModC   ;   ! Moderator temperature (C)
REAL BUnextSS ;
```

```
*****************************************************************
* END DECLARATION                                               *
*****************************************************************
XSM_FILE BURN32S ;
XSM_FILE BURN156S ;
INTEGER iFirst iLast ;
{% if mpi_enabled %}
MODULE DRVMPI: SNDMPI: ;
INTEGER cpumain := 0 ;
INTEGER cpurank totcpu ;
DRVMPI: :: EDIT 0 MY-ID >>cpurank<< WORLD-SIZE >>totcpu<<
           SETLOOP B0 {{ n_reprises }} >>iFirst<< >>iLast<< ;
{% endif %}
ECHO iFirst "=>" iLast ;
*****************************************************************
* BEGIN COMPO CREATION                                          *
*****************************************************************
{% for assembly in assembly_list %}
XSM_FILE LIBEQS ;
LINKED_LIST {{ assembly.LIB26G }} ;
{{ assembly.LIB26G }} := LIBEQS   ;
COMPO := COMPO: {{ assembly.LIB26G }} ::
  EDIT 0
  MAXCAL 5
  COMM 'evolution BURNUP Assembly' ENDC
  PARA 'TCOM' VALU REAL !TEMPERATURE_COMBUSTIBLE
  PARA 'DMOD' VALU REAL !DENSITE_MODERATEUR
  PARA 'CBOR' CONC B10 {{ assembly.LIB26G }} 4 !At_Bore10
  PARA 'NXEN' VALU REAL !NIV_XENON
  PARA 'BURN' VALU REAL !BURNUP
  INIT
  ;
{{ assembly.LIB26G }} := DELETE: {{ assembly.LIB26G }} ;
{% endfor %}
*****************************************************************
* END COMPO CREATION                                            *
*****************************************************************
*----
*  Boucle sur les conditions
*----
INTEGER nTcom nCBor nTmod nMaBU := 3 5 5 28 ;
INTEGER iTcom iCBor iTmod iMaBU ;
INTEGER idens ;

INTEGER iCount := iFirst ;
{% for assembly in assembly_list %}
REAL Tend{{ assembly.tag }} BUend{{ assembly.tag }} ;
REAL {{ assembly.butarget }} := {{ assembly.burnup }} ;
INTEGER {{ assembly.istep }}
        {{ assembly.ibustep }}
        {{ assembly.iautop }}
;
LINKED_LIST {{ assembly.FLUX1 }}
            {{ assembly.FLUX2 }}
            {{ assembly.BURN156}}
```

```
            {{ assembly.LIBSUBG }}
            {{ assembly.LIB281G }}
;
WHILE iCount iLast <= DO
BURN32  := BURN32S  ;
BURN156 := BURN156S ;
EVALUATE {{ assembly.istep }}
         {{ assembly.ibustep }}
         {{ assembly.iautop }} := 0 0 0 ;
EVALUATE Tend{{ assembly.tag }} BUend{{ assembly.tag }}
         := 0.0 0.0 ;
EVALUATE iTcom := iCount nTcom % 1 + ;
EVALUATE iCBor := iCount nTcom / nCBor % 1 + ;
EVALUATE iTmod := iCount nTcom nCBor * / nTmod % 1 + ;
EVALUATE idens := iTmod ;
ECHO iCount "," iTmod "," iCBor "," iTcom ;
GREP: DONNEES :: GETVAL 'TCOM' <<iTcom>> >>TfuelC<< ;
GREP: DONNEES :: GETVAL 'CBOR' <<iCBor>> >>pbore<< ;
GREP: DONNEES :: GETVAL 'Tmod' <<iTmod>> >>TModC<< ;
GREP: DONNEES :: GETVAL 'DMOD' <<idens>> >>dens_mod<< ;
*******************************************************************
* BEGIN LIBRARY READING                                           *
*******************************************************************
{{ assembly.LIBSUBG }} := Mix_UOX_{{ NbRegions }} ::
  <<dens_mod>> <<pbore>> <<N_U5_UOX>> <<densU_UOX>>
  <<TfuelC>> <<TModC>>
;
{{ assembly.LIB281G }} Autop1
{{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
*******************************************************************
* END LIBRARY READING                                             *
*******************************************************************
{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
evo_ass_depl
{{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
'{{ assembly.LIB26G }}'
<<NbRegions>>
<<{{ assembly.istep }}>>
;
EDI2G := Cond_2gr_SPH {{ assembly.FLUX2 }}
                      {{ assembly.LIB26G }}
                      TRACKN2 :: <<Library>>
;
{{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
{{ assembly.BURN156 }} {{ assembly.LIB26G }}
{{ assembly.FLUX2 }} TRACKN2 ::
  EDIT 0
  SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
;

COMPO := COMPO: COMPO EDI2G
{{ assembly.BURN156 }} {{ assembly.LIB26G }} ::
  EDIT 0
  MACRO
  SET <<Tend{{ assembly.tag }}>> DAY
```

```
  'TCOM' <<TfuelC>>
  'DMOD' <<dens_mod>>
  'NXEN' <<n_xenon>>
  'BURN' <<BUend{{ assembly.tag }}>>
;
EVALUATE {{ assembly.istep }}
         {{ assembly.ibustep }}
         {{ assembly.iautop }} := 1 1 2 ;
GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>> >>BUnextSS<< ;
ECHO "BUnextSS" BUnextSS ;
EVALUATE iMaBU := 1 ;
WHILE iMaBU 30 < DO
  EVALUATE iMaBU := iMaBU 1 + ;
  EDI2G {{ assembly.LIB26G }} := DELETE:
  EDI2G {{ assembly.LIB26G }} ;
  GREP: DONNEES :: GETVAL 'MaBU' <<iMaBU>> >>{{ assembly.ibustep }}<< ;
  EVALUATE {{ assembly.istep }} := {{ assembly.ibustep }} ;
  GREP: DONNEES :: GETVAL 'burn'
               <<{{ assembly.ibustep }}>> >>BUend{{ assembly.tag }}<< ;
  EVALUATE Tend{{ assembly.tag }} := BUend{{ assembly.tag }} Fuelpwr / ;
*----
*  MAJ de la librairie de 1er niveau avec les conc de l'evo nom.
*----
   {{ assembly.LIB281G }} := LIB: {{ assembly.LIB281G }} BURN32 ::
    EDIT 0
    BURN <<{{ assembly.istep }}>>
! pas de donnes pour les  mis precedent pour ne pas changer [B10]
    MIX  9 MIX 10 MIX 11 MIX 12 MIX 13 MIX 14 MIX 15 MIX 16
    MIX 17 MIX 18 MIX 19 MIX 20 MIX 21 MIX 22 MIX 23 MIX 24
    MIX 25 MIX 26 MIX 27 MIX 28 MIX 29 MIX 30 MIX 31 MIX 32
    MIX 33 MIX 34 MIX 35 MIX 36 MIX 37 MIX 38 MIX 39 MIX 40
  ;
  {{ assembly.LIB281G }} := LIB:
  {{ assembly.LIB281G }} :: EDIT 0 MACR MIXS
  ;

  ECHO "BURNUP_sortie = " BUend{{ assembly.tag }} "Next SS = " BUnextSS ;
  ECHO BUend{{ assembly.tag }} BUnextSS - ABS ;
  IF BUend{{ assembly.tag }} BUnextSS - ABS 1E-5 < THEN
    {{ assembly.LIB281G }} Autop2
    {{ assembly.LIB281G }}
    {{ assembly.LIBSUBG }} TRACKSS :: <<NbRegions>> ;
    EVALUATE {{ assembly.iautop }} := {{ assembly.iautop }} 1 + ;
    IF {{ assembly.iautop }} 10 <= THEN
      GREP: DONNEES :: GETVAL 'autop' <<{{ assembly.iautop }}>>
                                      >>BUnextSS<< ;
    ENDIF ;
  ENDIF ;

  {{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  evo_ass_depl          {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
  {{ assembly.LIB281G }} {{ assembly.BURN156 }} TRACKN1 TRACKN2 ::
  '{{ assembly.LIB26G }}'
  <<NbRegions>>
  <<{{ assembly.istep }}>>
```

```
  ;

  {{ assembly.BURN156 }} {{ assembly.LIB26G }} := EVO:
  {{ assembly.BURN156 }} {{ assembly.LIB26G }}
  {{ assembly.FLUX2 }} TRACKN2 ::
    EDIT 0
    SAVE <<Tend{{ assembly.tag }}>> DAY POWR <<Fuelpwr>>
  ;

  EDI2G := Cond_2gr_SPH {{ assembly.FLUX2 }}
                        {{ assembly.LIB26G }}
                        TRACKN2 :: <<Library>>
  ;

  COMPO := COMPO: COMPO EDI2G
  {{ assembly.BURN156 }} {{ assembly.LIB26G }} ::
    EDIT 0
    MACRO
    SET <<Tend{{ assembly.tag }}>> DAY
     'TCOM' <<TfuelC>>
     'DMOD' <<dens_mod>>
     'NXEN' <<n_xenon>>
     'BURN' <<BUend{{ assembly.tag }}>>
  ;
ENDWHILE ; ! WHILE iMaBU 28 < DO
EVALUATE iCount := iCount 1 + ;
{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
:= DELETE:
{{ assembly.LIB26G }} {{ assembly.FLUX1 }} {{ assembly.FLUX2 }}
;
{{ assembly.LIBSUBG }} {{ assembly.LIB281G }} := DELETE:
{{ assembly.LIBSUBG }} {{ assembly.LIB281G }} ;
EDI2G BURN32 BURN156 := DELETE:
EDI2G BURN32 BURN156 ;
ENDWHILE ; !iCount iLast <=
{% endfor %}

XSM_FILE MULTICOMPO ;
MULTICOMPO := COMPO ;

TRACKSS TRACKN1 TRACKN2 := DELETE:
TRACKSS TRACKN1 TRACKN2 ;

END: ;
QUIT .
```

Le script CLE-2000 du calcul de campagne avec bibliothèque :

```
*----------------------------------------------------------------------
*
*Purpose:
* MULTICOMPO campaign length calculation
*
```

```
*Copyright:
* Copyright (C) 2011 Ecole Polytechnique de Montreal
* This library is free software; you can redistribute it and/or
* modify it under the terms of the GNU Lesser General Public
* License as published by the Free Software Foundation; either
* version 2.1 of the License, or (at your option) any later version.
*
*Author(s): B. Toueg
*
*----------------------------------------------------------------------
*  Contrat EDF/EPM
*  Nom          : GR514ben.x2m
*  Type         : fichier DONJON
*  Auteur       : B. Toueg
*  Date         : 2009/10/14
*  Reference    : GRAVELINES 514 campaign (2006).
MODULE DELETE: ABORT: END: ;

REAL pbore := 500. ;
REAL coeff := 0.199 6.022E+23 1.0E-06 * * 10.8110164 1.0E+24 * / ;
REAL N_B10 ;
************************************************************************
**************************             *******************************
*************                                      *****************
*                          DONJON INIT
*************                                      *****************
**************************             *******************************
************************************************************************
PROCEDURE core_ ;
LINKED_LIST COREGEO ;
INTEGER MaxR ;
COREGEO := core_ :: >>MaxR<< ;
MODULE USPLIT: ;
LINKED_LIST COREMATEX ;

COREGEO COREMATEX := USPLIT: COREGEO :: NGRP 2 MAXR <<MaxR>>
                     NREFL 3  RMIX 1 2 3
                     NFUEL {{ fuel_mix|length }}  FMIX
{{ fuel_mix|join:' '|wordwrap:70 }}
;
MODULE TRIVAT: ;
LINKED_LIST CORETRACK ;
INTEGER MCFD := 3 ;
CORETRACK := TRIVAT: COREGEO :: MAXR <<MaxR>> MCFD <<MCFD>> ;
PROCEDURE fuelmap_init ;
LINKED_LIST COREFUELMAP ;
COREFUELMAP COREMATEX := fuelmap_init COREMATEX ;
PROCEDURE get_refl ;
LINKED_LIST MACNOEVO ;
MACNOEVO := get_refl ;
REAL Fuelpwr := 38.425 ; ! W/g ou kW/kg ou MW/t

************************************************************************
**************************             *******************************
*************                                      *****************
```

```
*                            DONJON CALCULATION
*************                                                 ****************
****************************              *******************************
************************************************************************
REAL Power := 2775. ; ! in MW
REAL fq ;
REAL mean_bu ;
REAL pbore_min pbore_max := 0. 1900. ;
REAL pbore_old ;
REAL keff keff_old ;
REAL eps_r ;
REAL burn_step := 150. ;
LOGICAL cbc_above_10 := $True_L ;
LOGICAL root_found go_brent ;
REAL x1 y1 x2 y2 y3 ;
INTEGER iter_burn iter_cbc ;
MODULE GREP: FIND0: FLPOW: TINST: RESINI: ;
MODULE RESINI: NCR: MACINI: TRIVAA: FLUD: ;
LINKED_LIST COREFLUX POWER BRENT ;
LINKED_LIST MACFL COREMACROEXT COREMATRIX ;
XSM_FILE XSMCPO :: FILE 'XSMCPO' ;
LINKED_LIST COMPO ;
COMPO := XSMCPO ;

EVALUATE iter_burn := 0 ;
WHILE iter_burn {{ iter_burn_max }} < DO
EVALUATE iter_burn := iter_burn 1 + ;
*----
*  CBC LOOP
*----
IF iter_burn 1 = THEN
  EVALUATE burn_step :=  150. ;
ELSEIF iter_burn 2 = THEN
  EVALUATE burn_step :=  850. ;
ELSE ! iter_burn > 2
  EVALUATE burn_step := 1000. ;
ENDIF ;

GREP: COREFUELMAP :: MEAN 'BURN-INST    ' 1 * >>mean_bu<< ;
ECHO "mean_bu" mean_bu ;
EVALUATE keff eps_r := 10. 1E-6 ;
EVALUATE iter_cbc := 0 ;
EVALUATE root_found := $False_L ;
WHILE root_found NOT DO
  EVALUATE iter_cbc := iter_cbc 1 + ;
  IF iter_cbc 1 = THEN
    EVALUATE pbore := pbore_min ;
  ELSEIF iter_cbc 2 = THEN
    EVALUATE pbore := pbore_max ;
  ENDIF ;

  EVALUATE N_B10 := coeff pbore * ;
COREFUELMAP  := RESINI: COREFUELMAP ::
  EDIT 0
  SET-PARAM 'CBOR' TIMES 'DMOD' SAME <<N_B10>>
```

```
  ;
MACFL := NCR: COMPO COREFUELMAP
      :: EDIT 0
         MACRO
         LINEAR
         TABLE COMPO default 'BURN'
               MIX 4 FROM 1
               MICRO ALL
               ENDMIX
;
COREMACROEXT COREMATEX := MACINI: COREMATEX MACNOEVO MACFL :: EDIT 1 ;
COREMATRIX := TRIVAA: COREMACROEXT CORETRACK :: EDIT 1 ;
IF iter_cbc 1 = THEN
  COREFLUX := FLUD: COREMATRIX CORETRACK ::
    EDIT 2 ACCE 4 3 EXTE 1E-6 ADI 5 ;
ELSE
  COREFLUX := FLUD: COREFLUX COREMATRIX CORETRACK ::
    EDIT 2 ACCE 4 3 EXTE 1E-6 ADI 5 ;
ENDIF ;
GREP: COREFLUX :: GETVAL 'K-EFFECTIVE ' 1  >>keff<<  ;

  EVALUATE root_found := 1. keff - ABS eps_r < ;

  ECHO "pbore       = " pbore ;
  ECHO "K-effective = " keff  ;

  IF root_found NOT THEN
    MACFL COREMACROEXT COREMATRIX := DELETE:
    MACFL COREMACROEXT COREMATRIX ;
    IF iter_cbc 1 = THEN
      EVALUATE pbore_old := pbore ;
      EVALUATE keff_old := keff ;
    ELSEIF iter_cbc 2 = THEN
      EVALUATE x1 y1 := pbore_old 1. keff_old - ;
      EVALUATE x2 y2 := pbore 1. keff - ;
      EVALUATE go_brent := y1 y2 * 0. < ;
      IF go_brent THEN
        BRENT := FIND0: ::
          POINT X <<x1>> Y <<y1>>
          POINT X <<x2>> Y <<y2>>
          >>root_found<< >>pbore<<
        ;
      ELSE
        EVALUATE root_found := $True_L ;
        EVALUATE pbore keff := 0. 0. ;
      ENDIF ;
    ELSE
      EVALUATE y3 := 1. keff - ;
      BRENT := FIND0: BRENT ::
        Y <<y3>>
        >>root_found<< >>pbore<<
      ;
    ENDIF ;
  ENDIF ;
```

```
ENDWHILE ; ! cbc loop

EVALUATE cbc_above_10 := pbore 10. > iter_burn {{ iter_burn_max }} < * ;

ECHO iter_cbc "iterations in the CBC loop gives :" ;
ECHO "Average Bu :       " mean_bu " MWd/t" ;
ECHO "Burn up increment :" burn_step " MWd/t" ;
ECHO "CBC :              " pbore "ppm" ;

IF cbc_above_10 THEN
  POWER := FLPOW: COREFUELMAP COREFLUX CORETRACK COREMATEX ::
    EDIT 0
    PTOT <<Power>>
  ;
  GREP: POWER :: GETVAL 'FORM-BUND    ' 1  >>fq<<  ;
  COREFUELMAP := TINST: COREFUELMAP POWER ::
    EDIT 0
    BURN-STEP <<burn_step>>
  ;
  ECHO "FQ :               " fq ;
*   COREFUELMAP := RESINI: COREFUELMAP :: INST-BVAL SMOOTH ;
  POWER := DELETE: POWER ;
  IF iter_cbc 2 > THEN
    BRENT := DELETE: BRENT ;
  ENDIF ;
  {% for suffix in xsmSuffixes %}
  IF iter_burn {{ suffix|floatformat }} = THEN
  XSM_FILE xsmFMAP{{suffix}} ;
  xsmFMAP{{suffix}} := COREFUELMAP ;
  XSM_FILE xsmFLUX{{suffix}} ;
  MODULE OUT: ;
  xsmFLUX{{suffix}} := OUT: COREFLUX COREMACROEXT CORETRACK COREGEO ::
         EDIT 0
         POWR <<Power>>
         INTG IN
  ;
  ENDIF ;
{% endfor %}
ELSE ! cbc_above_10 NOT
  ECHO "Campaign length terminated" ;
  END: ;
ENDIF ; ! cbc_above_10


MACFL COREMACROEXT COREMATRIX COREFLUX := DELETE:
MACFL COREMACROEXT COREMATRIX COREFLUX ;
ENDWHILE ;

END: ;
QUIT .
```